Questa® SIM Command Reference Manual

Including Support for Questa SV/AFV

Software Version 10.4c

© 1991-2015 Mentor Graphics Corporation
All rights reserved.

This document contains information that is proprietary to Mentor Graphics Corporation. The original recipient of this document may duplicate this document in whole or in part for internal business purposes only, provided that this entire notice appears in all copies. In duplicating any part of this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use and distribution of the proprietary information.

This document is for information and instruction purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

U.S. GOVERNMENT LICENSE RIGHTS: The software and documentation were developed entirely at private expense and are commercial computer software and commercial computer software documentation within the meaning of the applicable acquisition regulations. Accordingly, pursuant to FAR 48 CFR 12.212 and DFARS 48 CFR 227.7202, use, duplication and disclosure by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in the license agreement provided with the software, except for provisions which are contrary to applicable mandatory federal laws.

TRADEMARKS: The trademarks, logos and service marks ("Marks") used herein are the property of Mentor Graphics Corporation or other parties. No one is permitted to use these Marks without the prior written consent of Mentor Graphics or the owner of the Mark, as applicable. The use herein of a third-party Mark is not an attempt to indicate Mentor Graphics as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A current list of Mentor Graphics' trademarks may be viewed at: www.mentor.com/trademarks.

The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: www.mentor.com
SupportNet: supportnet.mentor.com/

Send Feedback on Documentation: supportnet.mentor.com/doc_feedback_form

Table of Contents

Chapter 1
Syntax and Conventions 15

Documentation Conventions 15
File and Directory Pathnames 16
Design Object Names 16
Object Name Syntax 16
Tcl Syntax and Specification of Array Bits and Slices 18
SystemC Class, Structure, and Union Member Specification 18
SystemVerilog Scope Resolution Operator 19
Specifying Names 20
Environment Variables and Pathnames 22
Name Case Sensitivity 23
Extended Identifiers 23
Wildcard Characters 23
Supported Commands 23
Using the WildcardFilter Preference Variable 24
Simulator Variables 28
Simulation Time Units 28
Optionsets 29
Argument Files 30
Command Shortcuts 31
Command History Shortcuts 32
Numbering Conventions 33
VHDL Numbering Conventions 33
Verilog Numbering Conventions 34
GUI_expression_format 34
Expression Typing 35
Expression Syntax 36
Signal and Subelement Naming Conventions 42
Grouping and Precedence 42
Concatenation of Signals or Subelements 43
Record Field Members 44
Searching for Binary Signal Values in the GUI 45

Chapter 2
Commands 47

.main clear 67
abort 68
add atv 69
add button 70
add dataflow 73
add list 75

add memory 80
add message 82
add schematic 84
add testbrowser 86
add watch 87
add wave 89
add_cmdhelp 97
add_menu 99
add_menucb 101
add_menuitem 103
add_separator 105
add_submenu 106
alias 108
archive load 109
archive write 110
assertion action 111
assertion active 114
assertion count 116
assertion enable 118
assertion fail 121
assertion pass 123
assertion profile 125
assertion report 127
atv log 129
batch_mode 131
bd 132
bookmark add wave 134
bookmark delete wave 136
bookmark goto wave 137
bookmark list wave 138
bp 139
call 146
cd 150
cdbg 151
change 154
change_menu_cmd 157
check contention add 158
check contention config 160
check contention off 161
check float add 162
check float config 164
check float off 165
check stable off 166
check stable on 167
checkpoint 169
classinfo ancestry 171
classinfo descriptive 172
classinfo find 174
classinfo implements 175

classinfo instances 177
classinfo interfaces 179
classinfo isa 181
classinfo report 182
classinfo stats 184
classinfo trace 185
classinfo types 187
compare add 189
compare annotate 195
compare clock 197
compare configure 199
compare continue 201
compare delete 202
compare end 203
compare info 204
compare list 206
compare options 207
compare reload 212
compare reset 213
compare run 214
compare savediffs 215
compare saverules 216
compare see 217
compare start 219
compare stop 221
compare update 222
configure 223
context 229
coverage analyze 233
coverage attribute 242
coverage clear 247
coverage create 249
coverage edit 252
coverage exclude 257
coverage goal 267
coverage loadtestassoc 271
coverage open 272
coverage ranktest 273
coverage report 281
coverage save 295
coverage tag 299
coverage testnames 306
coverage unlinked 307
coverage weight 311
dataset alias 314
dataset clear 315
dataset close 316
dataset config 317
dataset current 319

dataset info 320
dataset list 321
dataset open. 322
dataset rename. 323
dataset restart 324
dataset save 325
dataset snapshot 326
delete 329
describe 330
disablebp 332
disable_menu 333
disable_menuitem 334
do. 335
down 337
drivers 339
dumplog64 341
echo 342
edit. 343
enablebp 344
enable_menu 345
enable_menuitem 346
encoding 347
environment 348
examine 350
exit. 358
fcover configure 359
find 362
find connections 368
find drivers 369
find infiles 374
find insource 375
force 377
formatTime 383
fsm list. 384
fsm properties 385
fsm view 386
gc configure 387
gc run 389
gdb dir 390
getactivecursortime 391
getactivemarkertime 392
help 393
history 394
jobspy 395
layout. 396
lecho 398
left 399
log 402
lshift 406

lsublist 407
mem compare 408
mem display 409
mem list 412
mem load 413
mem save 417
mem search 420
next 423
noforce 424
nolog 425
notepad 427
noview 428
nowhen 429
onbreak 430
onElabError 432
onerror 433
onfinish 435
pa autotestplan 436
pa msg 437
pa report 440
pause 441
pop 442
power add 443
power off 445
power on 447
power report 449
power reset 452
precision 454
printenv 455
process report 456
profile clear 457
profile interval 458
profile off 459
profile on 460
profile open 462
profile option 463
profile reload 465
profile report 466
profile save 470
profile summary 472
project 474
property list 477
property wave 479
push 481
pwd 482
questasim 483
quietly 484
quit 485
qverilog 486

radix 490
radix define 492
radix delete 496
radix list 497
radix names 498
radix signal 499
readers 500
report 501
restart 503
restore 505
resume 506
right 507
run 510
runStatus 513
sccom 515
scgenmod 525
sdfcom 528
search 530
searchlog 533
see 536
seetime 537
setenv 539
shift 540
show 541
simstats 542
simstatslist 544
stack down 546
stack frame 547
stack level 548
stack tb 549
stack up 550
status 551
step 552
stop 554
suppress 555
tb 557
tcheck_set 558
tcheck_status 562
Time 565
toggle add 568
toggle disable 571
toggle enable 573
toggle report 574
toggle reset 576
tr color 577
tr order 580
tr uid 582
transcribe 584
transcript 585

transcript file 586
transcript path 588
transcript sizelimit 589
triage dbfile 590
triage dbinsert 595
triage dbrefresh 597
triage query 598
triage report 601
triage view 604
tssi2mti 606
typespec 607
ui_VVMode 608
unsetenv 610
up 611
uvm call 613
uvm configtracing 616
uvm displayobjections 617
uvm findregisters 620
uvm findsequences 622
uvm handle 624
uvm mapmode 626
uvm printconfig 627
uvm printfactory 629
uvm printstreams 631
uvm printtopology 633
uvm setverbosity 635
uvm simpath 637
uvm traceobjections 639
uvm uvmpath 641
vcd add 643
vcd checkpoint 645
vcd comment 646
vcd dumpports 647
vcd dumpportsall 650
vcd dumpportsflush 651
vcd dumpportslimit 652
vcd dumpportsoff 653
vcd dumpportson 654
vcd file 655
vcd files 657
vcd flush 659
vcd limit 660
vcd off 661
vcd on 662
vcd2wlf 663
vcom 665
vcover attribute 691
vcover diff 696
vcover dump 701

vcover history 704
vcover merge. 707
vcover parallelmerge. 716
vcover ranktest 719
vcover remove. 728
vcover report 730
vcover stats 752
vcover testnames. 757
vdel 761
vdir 763
vencrypt. 766
verror. 770
vgencomp 772
vhencrypt. 774
view. 777
virtual count 781
virtual define 782
virtual delete 783
virtual describe 784
virtual expand 785
virtual function 786
virtual hide 790
virtual log 791
virtual nohide 793
virtual nolog 794
virtual region. 796
virtual save 797
virtual show. 798
virtual signal 799
virtual type 803
vlib 805
vlog 808
vmake 844
vmap 846
vopt 848
vsim. 906
vsim<info> 969
vsim_break 970
vsource 971
wave 972
wave create 976
wave edit. 982
wave export. 986
wave import 988
wave modify 989
wave sort. 994
when 995
where. 1003
wlf2log 1004

wlf2vcd 1006
wlfman 1007
wlfrecover 1014
write cell_report 1015
write format 1016
write list 1019
write preferences 1020
write report 1021
write timing 1025
write transcript 1027
write tssi 1028
write wave 1030
xml2ucdb 1032
xprop assertlimit 1039
xprop disable 1040
xprop enable 1041

Index

End-User License Agreement

List of Examples

Example 1-1. Base- and Descendant-Class Specification 19
Example 1-2. SystemVerilog Scope Resolution Operator Example 20
Example 1-3. Defining and Calling Optionsets 30

List of Figures

Figure 2-1. drivers Command Results in Transcript 339
Figure 2-2. find infiles Example 374
Figure 2-3. find insource Example. 376
Figure 2-4. readers Command Results in Transcript 500
Figure 2-5. UVM Sequences 623

List of Tables

Table 1-1. Conventions for Command Syntax 15
Table 1-2. Examples of Object Names 22
Table 1-3. Wildcard Characters in HDL Object Names 24
Table 1-4. WildcardFilter Arguments 26
Table 1-5. WildcardFilter Argument Groups 27
Table 1-6. Keyboard Shortcuts for Command History 32
Table 1-7. VHDL Number Conventions: Style 1 33
Table 1-8. VHDL Number Conventions: Style 2 33
Table 1-9. Verilog Number Conventions 34
Table 1-10. Constants Supported for GUI Expressions 37
Table 1-11. Array Constants Supported for GUI Expressions 37
Table 1-12. Variables Supported for GUI Expressions 37
Table 1-13. Array Variables Supported for GUI Expressions 38
Table 1-14. Operators Supported for GUI Expressions 39
Table 1-15. Precedence of GUI Expression Operators 40
Table 1-16. Casting Conversions Supported for GUI Expressions 41
Table 1-17. VHDL Logic Values Used in GUI Search 45
Table 1-18. Verilog Logic Values Used in GUI Search 46
Table 2-1. Supported Commands 47
Table 2-2. Message Viewer Categories 82
Table 2-3. Comparing Reference Objects to Test Objects 189
Table 2-4. Order and Type of Ranked Tests 273
Table 2-5. profile summary Header Rules for Return 473
Table 2-6. runStatus Command States 513
Table 2-7. runStatus -full Command Information 513
Table 2-8. Field Arguments for Window Searches 531
Table 2-9. Output Fields for tcheck_status Command 563
Table 2-10. Alias for Tables 598
Table 2-11. Warning Message Categories for vcom -nowarn 682
Table 2-12. Order and Type of Ranked Tests 719
Table 2-13. Design Unit Properties 764
Table 2-14. Warning Message Categories for vlog -nowarn 830
Table 2-15. vopt Arguments for Access Visibility Being Replaced 852
Table 2-16. Warning Message Categories for vopt -nowarn 881
Table 2-17. Wave Window Commands for Cursor 972
Table 2-18. Wave Window Commands for Expanded Time Display 972
Table 2-19. Wave Window Commands for Controlling Display 973
Table 2-20. Wave Window Commands for Zooming 973

Chapter 1
Syntax and Conventions

This manual uses the following conventions to define Questa SIM command syntax.

Note

Documentation for Questa SIM also supports Questa SV/AFV products. For more complete information on current support for Questa SV/AFV, refer to the Installation and Licensing Guide.

Documentation Conventions

The following conventions are used to define ModelSim command syntax

Table 1-1. Conventions for Command Syntax

Syntax notation	Description
< >	angled brackets surrounding a syntax item indicate a user-defined argument; do not enter the brackets in commands
[ ]	square brackets generally indicate an optional item; if the brackets surround several words, all must be entered as a group; the brackets are not entered[1]
{ }	braces indicate that the enclosed expression contains one or more spaces yet should be treated as a single argument, or that the expression contains square brackets for an index; for either situation, the braces are entered
…	an ellipsis indicates items that may appear more than once; the ellipsis itself does not appear in commands
|	the vertical bar indicates a choice between items on either side of it; do not include the bar in the command
monospaced type	monospaced type is used in command examples
#	comments included with commands are preceded by the number sign (#); useful for adding comments to DO files (macros)

1. One exception to this rule is when you are using Verilog syntax to designate an array slice. For example,

add wave {vector1[4:0]}

The square brackets in this case denote an index. The braces prevent the Tcl interpreter from treating the text within the square brackets as a Tcl command.

Note

Neither the prompt at the beginning of a line nor the <Enter> key that ends a line is shown in the command examples.

File and Directory Pathnames

Several Questa SIM commands have arguments that point to files or directories. For example, the **-y** argument to vlog specifies the Verilog source library directory to search for undefined modules.

Spaces in file pathnames must be escaped or the entire path must be enclosed in quotes. For example:

vlog top.v -y C:/Documents\ and\ Settings/projects/dut

or

vlog top.v -y "C:/Documents and Settings/projects/dut"

Design Object Names

Design objects are organized hierarchically, where various objects creates a new level in the hierarchy.

- **VHDL** — component instantiation statement, block statement, and package
- **Verilog** — module instantiation, named fork, named begin, task and function
- **SystemVerilog** — class, package, program, and interface
- **SystemC** — module instantiation

Object Name Syntax

There is a specific syntax for specifying object names in Questa SIM.

```
[<dataset_name><datasetSeparator>][<pathSeparator>][<hierarchicalPath>]
<objectName>[<elementSelection>]
```

where

- **dataset_name** — is the name mapped to the WLF file in which the object exists. The currently active simulation is the "sim" dataset. Any loaded WLF file is referred to by the logical name specified when the WLF file was loaded. Refer to the chapter "Recording Simulation Results With Datasets" in the User's Manual for more information.
- **datasetSeparator** — is the character used to terminate the dataset name. The default is colon (:), though a different character (other than backslash (\ )) may be specified as the dataset separator via the DatasetSeparator variable in the *modelsim.ini* file. This character must be different than the pathSeparator character.
- **pathSeparator** — is the character used to separate hierarchical object names. Normally, a backslash (\ ) is used for VHDL and a period (.) is used for Verilog, although other characters (except a backslash (\ )) may be specified via the PathSeparator variable in the *modelsim.ini* file. This character must be different than the datasetSeparator.

  Both (.) and forward slash (/) can be used for SystemC.

  Neither (.) nor (/) can be used when referring to the contents of a SystemVerilog package or class.
- **hierarchicalPath** — is a set of hierarchical instance names separated by a path separator and ending in a path separator prior to the objectName. For example, */top/proc/clk*.
- **objectName** — is the name of an object in a design.
- **elementSelection** — indicates some combination of the following:
  - **Array indexing** — Single array elements are specified using either parentheses (()) or square brackets ([]) around a single number. You must also surround the object and specified array element with curly braces ({}). Refer to Tcl Syntax and Specification of Array Bits and Slices for important information about using square brackets and parentheses in Questa SIM commands.
  - **Array slicing** — Slices (or part-selects) of arrays are specified using either parentheses (()) or square brackets ([]) around a range specification. A range is two numbers separated by one of the following: " to ", " downto ", or a colon (:). You must also surround the object and specified array slice with curly braces ({}). Refer to Tcl Syntax and Specification of Array Bits and Slices for important information about using square brackets and parentheses in Questa SIM commands.
  - **Record field selection** — A record field is specified using a period (.) followed by the name of the field.
  - C++ **class, structure, and union member selection** — A class, structure, or union member is specified using the record field specification syntax, described just above.

Tcl Syntax and Specification of Array Bits and Slices

Because Questa SIM is a Tcl-based tool, you must surround objects and signals with curly braces
({}) when specifying array bits or slices with parentheses (()), spaces, or square brackets ([]).

For example:

```
toggle add {data[3:0]}
toggle add {data(3 to 0)}
force {bus1[1]} 1
```

Further Details

Because Questa SIM is based on Tcl, its commands follow Tcl syntax. One problem you may encounter with Questa SIM commands is the use of square brackets ([]), parentheses (()), or spaces when specifying array bits and slices. As shown on the previous page, square brackets are used to specify slices of arrays (for example, *data[3:0]*). However, in Tcl, square brackets signify command substitution. Consider the following example:

```
set aluinputs [find -in alu/*]
```

Questa SIM evaluates the **find** command first and then sets variable *aluinputs* to the result of the find command. Obviously you don't want this type of behavior when specifying an array slice, so you would use curly brace escape characters:

```
add wave {/s/abc/data_in[10:1]}
```

You must also use the escape characters if using VHDL syntax with spaces:

```
add wave {/s/abc/data_in(10 downto 1)}
```

For complete details on Tcl syntax, refer to Tcl Command Syntax.

SystemC Class, Structure, and Union Member Specification

You can specify members of SystemC structures and classes using HDL record syntax.

The syntax for specifying members of a base class using Questa SIM is different than C++. In C++, it is not necessary to specify the base class:

```
<instance>.<base_member>
```

Whereas, in Questa SIM you *must* include the name of the base class:

```
<instance>.<base>.<base_member>
```

Example 1-1. Base- and Descendant-Class Specification

Let's say you have a base class and a descendant class:

```
class dog
{
   private:
   int value;
};
class beagle : public dog
{
   private:
   int value;
   dog d;
};
```

You have an sc_signal<> of type beagle somewhere in your code:

```
sc_signal<beagle> spot;
```

Legal names for viewing this signal are:

```
spot
spot.*
spot.value
spot.dog
spot.dog.*
spot.dog.value
```

Now, to examine the member *value* of the base class *dog*, you would type:

```
exa spot.dog.value
```

To examine the member *value* of member *d*, you would type:

```
exa spot.d.value
```

To examine the member *value*, you would type:

```
exa spot.value
```

SystemVerilog Scope Resolution Operator

SystemVerilog offers the scope resolution operator, double colon (::), for accessing classes within a package and static data within a class. The example below shows various methods of using this operator as well as alternatives using standard hierarchical references.

Example 1-2. SystemVerilog Scope Resolution Operator Example

```
package myPackage;
   class packet;
      static int a[0:1] = {1, 2};
      int b[0:1];
      int c;

      function new;
         b[0] = 3;
         b[1] = 4;
         c = a[0];
      endfunction
   endclass
endpackage : myPackage

module top;
   myPackage::packet my = new;
   int myint = my.a[1];
endmodule
```

The following examine examples access data from the class *packet*.

```
examine myPackage::packet::a
examine /top/my.a
```

Both of the above commands return the contents of the static array *a* within class *packet*.

```
examine myPackage::packet::a(0)
examine /top/my.a(0)
```

Both of the above commands return the contents of the first element of the static array *a* within class *packet*.

```
examine /top/my.b
```

Return the contents of the instance-specific array *b*.

```
examine /top/my.b(0)
```

Return the contents of the first element of the instance-specific array *b*.

When referring to the contents of a package or class, you cannot use the standard path separators, a period (.) or a forward slash (/).

Specifying Names

We distinguish between four "types" of object names: simple, relative, fully-rooted, and absolute.

- **Simple name** — does not contain any hierarchy. It is simply the name of an object (e.g., *clk* or *data[3:0]*) in the current context.

- **Relative name** — does not start with a path separator and may or may not include a dataset name or a hierarchical path (e.g., *u1/data* or *view:clk*). A relative name is relative to the current context in the current or specified dataset.
- **Fully-rooted name** — starts with a path separator and includes a hierarchical path to an object (e.g., */top/u1/clk*).There is a special case of a fully-rooted name where the top-level design unit name can be unspecified (e.g., */u1/clk*). In this case, the first top-level instance in the design is assumed.
- **Absolute name** — is an exactly specified hierarchical name containing a dataset name and a fully rooted name (e.g., *sim:/top/u1/clk*).

The current dataset is used when accessing objects where a dataset name is not specified as part of the name. The current dataset is determined by the dataset currently selected in the Structure window or by the last dataset specified in an environment.

The current context in the current or specified dataset is used when accessing objects with relative or simple names. The current context is either the current process, if any, or the current instance if there is no current process, or the current process is not in the current instance. The situation of the current process not being in the current instance can occur, for example, by selecting a different instance in the Structure tab or by using the environment to set the current context to a different instance.

The current context is also the activation level of an automatic task, function, or block. Different levels of activation may be selected by using the Call Stack window, or by using the 'stack up' or 'stack down' commands.

For example, when you set a breakpoint on line 5 of the following code:

```
package p;
   int I;
   function automatic int factorial(int n);
      if(n==0)
    return 1;
      else
    return  n * factorial(n - 1);
   endfunction : factorial
endpackage : p
module top;
   initial begin
      p::I=p::factorial(3);

      $display(p::I);
      $display(p::factorial(4));
   end
endmodule: top
```

When you issue the command:

examine n

the transcript returns:

```
0
```

However, when you issue the command:

```
stack up;examine n
```

the transcript returns:

```
1
```

Table 1-2 contains examples of various ways of specifying object names.

Table 1-2. Examples of Object Names

Object Name	Description
clk	specifies the object *clk* in the current context
/top/clk	specifies the object *clk* in the top-level design unit.
/top/block1/u2/clk	specifies the object *clk*, two levels down from the top-level design unit
block1/u2/clk	specifies the object *clk*, two levels down from the current context
array_sig[4]	specifies an index of an array object
{array_sig(1 to 10)}	specifies a slice of an array object in VHDL or SystemC; see Tcl Syntax and Specification of Array Bits and Slices for more information
{mysignal[31:0]}	specifies a slice of an array object in Verilog or SystemC; see Tcl Syntax and Specification of Array Bits and Slices for more information
record_sig.field	specifies a field of a record, a C++ class or structure member, or a C++ base class

Environment Variables and Pathnames

You can substitute environment variables for pathnames in any argument that requires a pathname.

For example:

```
vlog -v $lib_path/und1
```

Assuming you have defined $lib_path on your system, vlog will locate the source library file *und1* and search it for undefined modules. Refer to Environment Variables for more information.

Name Case Sensitivity

Name case sensitivity is different for VHDL and Verilog. VHDL names are not case sensitive except for extended identifiers in VHDL 1076-1993 or later. In contrast, all Verilog names are case sensitive.

Names in Questa SIM commands are case sensitive when matched against case sensitive identifiers, otherwise they are not case sensitive.

SystemC names are case sensitive.

Extended Identifiers

The following are supported formats for extended identifiers for any command that takes an identifier.

```
{\ext ident!\ }
   # Note that trailing space before closing brace is required

\\ext\ ident\!\\
   # All non-alpha characters escaped
```

Wildcard Characters

You can use wildcard characters in HDL object names in many simulator commands.

Supported Commands

There are a number of commands that support wildcard characters.

The following is a partial list of the commands:

- add dataflow
- add list
- add memory
- add watch
- add wave
- describe
- dumpports
- examine
- find (see the Examples section in the find command for wildcard searches in foreach loops to be applied with commands that do not accept wildcards.)

- log
- vcd add

When you execute any of these commands with a wildcard, the default behavior is to exclude the following object types:

- VHDL shared variables in packages and design units, constants, generics, and immediate assertions
- Verilog parameters, specparams, memories
- SystemVerilog multi-dimensional arrays and class objects
- PSL and SystemVerilog assertions, covers, and endpoints
- Signals in cells
- Non-dynamic objects of a size equal to or greater than the level specified in the WildcardSizeThreshold *modelsim.ini* variable if the variable has been enabled. Refer to the WildcardSizeThreshold and WildcardSizeThresholdVerbose *modelsim.ini* variables for more information.

You can alter these exclusions with the WildcardFilter preference variable. Refer to the section "Using the WildcardFilter Preference Variable" for more information.

Table 1-3 identifies these supported wildcard characters.

Table 1-3. Wildcard Characters in HDL Object Names

Character Syntax	Description
*	matches any sequence of characters
?	matches any single character
[]	matches any one of the enclosed characters; a hyphen can be used to specify a range (for example, a-z, A-Z, 0-9); can be used *only* with the find command

Note

A wildcard character does not match a path separator. For example, */dut/** will match */dut/siga* and */dut/clk*. However, */dut** will not match either of those.

Using the WildcardFilter Preference Variable

The WildcardFilter preference variable controls which object types are excluded when performing wildcard matches with simulator commands. The WildcardFilter preference variable is a Tcl List and can be modified using Tcl commands.

The default object types are defined with the WildcardFilter *modelsim.ini* variable and load at each invocation of the simulator. You can add both individual (Table 1-4) and group objects (Table 1-5) to the current variable list, and you can remove individual objects from the current list.

Determining the Current WildcardFilter Variable Settings

Enter one of the following commands:

```
set WildcardFilter
```

or

```
echo $WildcardFilter
```

which returns the list of currently set variables.

Changing the WildcardFilter Settings from the Command Line

Refer to the list of WildcardFilter arguments in Table 1-4 and Table 1-5 to determine what you want to include in the wildcard matches.

- To define a new list of values enter the following command:

  ```
  set WildcardFilter “<arg1 arg2 ...>”
  ```

  Note that you must enclose the space-separated list of arguments in quotation marks.

- To add one or more values to the current list enter the following command:

  ```
  lappend WildcardFilter <arg1 arg2 ...>
  ```

  Note that you must not enclose the space-separated list of arguments in quotation marks.

- To remove a value from the filter use the set command with the Tcl lsearch command to create the new list from the existing list. For example:

  ```
  set WildcardFilter [lsearch -not -all -inline $WildcardFilter Endpoint]
  ```

Changing the WildcardFilter Settings back to the Default

Enter the following command:

```
set WildcardFilter default
```

Changing the WildcardFilter settings from the GUI

1. Choose **Tools** > **Wildcard Filter** from the main menu.
2. Select the individual Filters you want to exclude from wildcard searches (Table 1-4 describes each option), or select Composite Filters to activate related filters (Table 1-5 describes each composite option).

3. Click OK.

Refer to the Tcl man pages (**Help>Tcl Man Pages**) for more information about the lsearch and set commands.

Changing the default WildcardFilter settings

1. Open the modelsim.ini file for editing (refer to Making Changes to the modelsim.ini File)
2. Select the individual Filters you want to exclude from wildcard searches (Table 1-4 describes each option), or select Composite Filters to activate related filters (Table 1-5 describes each composite option).
3. Edit the WildcardFilter variable
4. Save the modelsim.ini file to your working directory.

WildcardFilter Argument Descriptions

Table 1-4 provides a list of the WildcardFilter arguments.

Table 1-4. WildcardFilter Arguments

Argument	Description
Alias	VHDL Alias
Assertion	Concurrent SystemVerilog or PSL assertion
CellInternal	Signals in cells, where a cell is defined as 1) a module within a ‘celldefine 2) a Verilog module found with a library search (using either vlog -v or vlog -y) and compiled with vlog +libcell or 3) a module containing a specify block
Class	Verilog class declaration
ClassReference	SystemVerilog class reference
Compare	Waveform comparison signal
Constant	VHDL constant
Cover	SystemVerilog or PSL cover statements
Covergroup	SystemVerilog or PSL covergroup
Coverpoint	Verilog coverpoint
Cross	Verilog cross
Endpoint	SystemVerilog assertion objects created for sequences on which the method “ended/triggered” is used. PSL assertion objects created for sequences for which the built in function “ended()” is used.

Table 1-4. WildcardFilter Arguments (cont.)

Argument	Description
Generic	VHDL generic
ImmediateAssert	VHDL immediate assertions
Integer	VHDL integer
Memory	Verilog memories
NamedEvent	Verilog named event
Net	Verilog net
Parameter	Verilog parameter
Real	Verilog real registers
Reg	Verilog register
ScVariable	SystemC variable
Signal	VHDL signal
SpecParam	Verilog specparam
Time	Verilog time registers
Transaction	Transaction stream and stream arrays
Variable	VHDL shared variables in packages and design units.
VHDLFile	VHDL files
VirtualExpr	Virtual expression
VirtualSignal	Virtual signal

Table 1-5 provides a list of the group aliases of WildcardFilter arguments. You can set a group value with the set command. The expanded list of values is returned.

Table 1-5. WildcardFilter Argument Groups

Group Argument	Specific arguments included
AllVHDL	Architecture, Block, Generate, Package, Foreign, Process, Signal, Variable, Constant, Generic, Alias, Subprogram, VHDLFile
AllVerilogVars	Parameter, Reg, Integer, Time, Real, SpecParam, Memory, NamedEvent, ClassReference
AllVerilog	Net, Parameter, Reg, Integer, Time, Real, SpecParam, Memory, NamedEvent, Class, Cross, Covergroup, Coverpoint, ClassReference
VirtualSignals	VirtualSignal, VirtualExpr

Table 1-5. WildcardFilter Argument Groups (cont.)

Group Argument	Specific arguments included
SystemC	ScVariable
AllHDLSignals	Signal, Variable, Constant, Generic, Alias, Net, Parameter, Reg, Integer, Time, Real, SpecParam, Memory, NamedEvent, VirtualSignal, VirtualExpr, ClassReference
AllVariables	Variable, Constant, Generic, Alias, Parameter, Reg, Integer, Time, Real, SpecParam, Memory, NamedEvent, ClassReference
AllHDLSignalsVars	Signal, Variable, Constant, Generic, Alias, Net, Parameter, Reg, Integer, Time, Real, SpecParam, Memory, NamedEvent, VirtualSignal, VirtualExpr, ClassReference
AllSignals	Signal, Net, Parameter, Reg, Integer, Time, Real, SpecParam, Memory, NamedEvent, VirtualSignal, VirtualExpr, Endpoint, ClassReference
AllSignalsVars	Signal, Variable, Constant, Generic, Alias, Net, Parameter, Reg, Integer, Time, Real, SpecParam, Memory, NamedEvent, VirtualSignal, VirtualExpr, Endpoint, ScVariable, ClassReference
AllConstants	Constant, Generic, Parameter, SpecParam
Default	Variable, Constant, Generic, Parameter, SpecParam, Memory, Assertion, Cover, Endpoint, ScVariable, CellInternal, ImmediateAssert VHDLFile

Simulator Variables

You can reference Questa SIM variables in a simulator command by preceding the name of the variable with the dollar sign ($) character.

Questa SIM uses global variables for simulator state variables, simulator control variables, simulator preference variables, and user-defined variables. Refer to modelsim.ini Variables in the User's Manual for more information on variables.

The report command returns a list of current settings for either the simulator state or simulator control variables.

Simulation Time Units

You can specify the time unit for delays in all simulator commands that have time arguments.

For example:

```
force clk 1 50 ns, 1 100 ns -repeat 1 us
run 2 ms
```

Note that all the time units in a Questa SIM command need not be the same.

Unless you specify otherwise as in the examples above, simulation time is always expressed using the resolution units that are specified by the UserTimeUnit variable.

By default, the specified time units are assumed to be relative to the current time unless the value is preceded by the character @, which signifies an absolute time specification.

Optionsets

By defining and calling optionsets, you can easily use and combine common command line options.

The executable expands these optionsets and passes them to the tool as if they appeared directly on the command line. The behavior is similar to the **-f <file>** option.

Defining an Optionset

Define your optionsets in the [DefineOptionset] section of the *modelsim.ini* file, where the syntax is:

```
<optionset_name> = <command_arguments>
```

- <optionset_name> — a string that begins with a letter, and contains only letters, numbers, or underscores. The name is case-insensitive.
- <command_arguments> — a list of arguments as you would specify them on the command line. This list of arguments can:
  - Refer to another <optionset_name>, enclosed in percent-signs (%).
  - Include shell environment variables, preceded by a dollar-sign ($). If you embed the variable in a string, you must surround it with parentheses.

You can instruct the executable to return all the values of any optionsets as they are read with the following entry in the [optionsets] section.

```
PRINT_OPTIONSET_VALUE = 1
```

Calling an Optionset

Call your defined optionsets with the **-optionset** argument to the commands: **vlog**, **vcom, sccom**, **vopt** and **vsim**.

The syntax of -optionset is:

```
<command> -optionset <optionset_name>
```

Example 1-3. Defining and Calling Optionsets

Given the *modelsim.ini* file contents:

```
[DefineOptionset]
UVMINCLUDES = +incdir+$UVM_HOME/src
UVMDEBUG = -uvmcontrol=all -msgmode both
DEBUGALL = -classdebug -assertdebug %UVMDEBUG%
VLOGUVM = -vopt %UVMINCLUDES%
```

you can execute the following commands (note that optionset names are case-insensitive):

```
vlog -optionset VLOGUVM top.sv
vsim -optionset debugall mytop
```

which would expand to:

```
vlog -vopt +incdir+/home/usern/libs/uvm-1.1b/src
vsim -classdebug -assertdebug -uvmcontrol=all -msgmode both
```

Argument Files

You can load additional arguments into some commands by using argument files, which are specified with the -f argument.

The following commands support the -f argument:

- sccom
- vlog
- vcom
- vencrypt
- vmake
- vopt
- vsim

The **-f <filename>** argument specifies a file that contains additional command line arguments. The following sections outline some syntax rules for argument files.

- Single Quotes (‘ ’) — Allows you to group arbitrary characters so that no character substitution occurs within the quotes, such as environment variable expansion or escaped characters.

  ```
  +acc=rn+'\mymodule'
  //does not treat the '\' as an escape character
  ```

- Double Quotes (“ ”) — Allows you to group arbitrary characters so that Tcl-style backslash substitution and environment variable expansion is performed.

```
+acc=rn+"\\mymodule\\$VAR"
// escapes the path separators (\) and substitues
// your value of '$VAR'
```

- Unquoted — The following are notes on what occurs when some information is not quoted:
  - Backslash substitution — Any unquoted backslash (\ ) will be treated as an escape character.

```
+acc=rn\\mymodule
// the leading '\' is considered an escape character
```

  - Environment variable expansion — Any unquoted environment variable, such as $envname, will be expanded. You can also use curly braces ({ }) in your environment variable, such as ${envname}.

```
+acc=rn\\$MODULE
// the leading '\' is considered an escape character and the
// variable $MODULE is expanded
```

- Newline Character — You can specify arguments on separate lines in the argument file with the line continuation character (\ ). You must use a space before the backslash.
- Comments — Comments within the argument files follow these rules:
  - All text in a line beginning with // to its end is treated as a comment.
  - All text bracketed by /* … */ is treated as a comment.
  - All text in a line beginning with # to its end is treated as a comment.

Command Shortcuts

The following shortcut techniques are available on the command line.

- You may abbreviate command syntax, but there’s a catch — the minimum number of characters required to execute a command are those that make it unique. Remember, as we add new commands some of the old shortcuts may not work. For this reason Questa SIM does not allow command name abbreviations in macro files. This minimizes your need to update macro files as new commands are added.
- Multiple commands may be entered on one line if they are separated by semi-colons (;). For example:

Questa SIM> vlog -nodebug=ports level3.v level2.v ; vlog -nodebug top.v

The return value of the last function executed is the only one printed to the transcript. This may cause some unexpected behavior in certain circumstances. Consider this example:

```
vsim -c -do "run 20 ; simstats ; quit -f" top
```

You probably expect the **simstats** results to display in the Transcript window, but they will not, because the last command is **quit -f**. To see the return values of intermediate commands, you must explicitly print the results. For example:

```
vsim -do "run 20 ; echo [simstats]; quit -f" -c top
```

Command History Shortcuts

You can review simulator command history or rerun previous commands by using keyboard shortcuts at the Questa SIM/VSIM prompt.

Table 1-6 contains a list of these shortcuts.

Table 1-6. Keyboard Shortcuts for Command History

Shortcut	Description
!!	repeats the last command
!n	repeats command number n; n is the VSIM prompt number (for example, for this prompt: VSIM 12>, n =12)
!<string>	shows a list of executed commands that start with <string>; Use the up and down arrows to choose from the list
!abc	repeats the most recent command starting with "abc"
^xyz^ab^	replaces "xyz" in the last command with "ab"
up arrow and down arrow keys	scrolls through the command history
Ctrl-N (UNIX only)	scroll to the next command
Ctrl-P (UNIX only	scroll to the previous command
click on prompt	left-click once on a previous Questa SIM or VSIM prompt in the transcript to copy the command typed at that prompt to the active cursor
his or history	shows the last few commands (up to 50 are kept)

Numbering Conventions

Numbers in Questa SIM can be expressed in either VHDL or Verilog style. You can use two styles for VHDL numbers and one for Verilog.

VHDL Numbering Conventions

There are two types of VHDL number styles:

VHDL Style 1

```
[ - ] [ radix # ] value [ # ]
```

Table 1-7. VHDL Number Conventions: Style 1

Element	Description
-	indicates a negative number; optional
radix	can be any base in the range 2 through 16 (2, 8, 10, or 16); by default, numbers are assumed to be decimal; optional
value	specifies the numeric value, expressed in the specified radix; required
#	is a delimiter between the radix and the value; the first # sign is required if a radix is used, the second is always optional

A ‘-’ can also be used to designate a "don’t care" element when you search for a signal value or expression in the List or Wave window. If you want the ‘-’ to be read as a "don’t care" element, rather than a negative sign, be sure to enclose the number in double quotes. For instance, you would type "-0110--" as opposed to -0110--. If you don’t include the double quotes, Questa SIM will read the ‘-’ as a negative sign. For example:

```
16#FFca23#
2#11111110
-23749
```

VHDL Style 2

```
base "value"
```

Table 1-8. VHDL Number Conventions: Style 2

Element	Description
base	specifies the base; binary: B, octal: O, hex: X; required
"value"	specifies digits in the appropriate base with optional underscore separators; default is decimal; required

For example:

```
B"11111110"
X"FFca23"
```

Searching for VHDL Arrays in the Wave and List Windows

Searching for signal values in the Wave or List window may not work correctly for VHDL arrays if the target value is in decimal notation. You may get an error that the value is of incompatible type. Since VHDL does not have a radix indicator for decimal, the target value may get misinterpreted as a scalar value. Prefixing the value with the Verilog notation 'd should eliminate the problem, even if the signal is VHDL.

Verilog Numbering Conventions

Verilog numbers are expressed in the style:

```
[ - ] [ size ] [ base ] value
```

Table 1-9. Verilog Number Conventions

Element	Description
-	indicates a negative number; optional
size	the number of bits in the number; optional
base	specifies the base; binary: ‘b or ‘B, octal: ‘o or ‘O, decimal: ‘d or ‘D, hex: ‘h or ‘H; optional
value	specifies digits in the appropriate base with optional underscore separators; default is decimal; required

A ‘-’ can also be used to designate a "don’t care" element when you search for a signal value or expression in the List or Wave windows. If you want the ‘-’ to be read as a "don’t care" element, rather than a negative sign, be sure to enclose the number in double quotes. For instance, you would type "-0110--" as opposed to 7'b-0110--. If you don’t include the double quotes, Questa SIM will read the ‘-’ as a negative sign. For example:

```
’b11111110                  8’b11111110
’Hffca23                    21’H1fca23
-23749
```

GUI_expression_format

The GUI_expression_format is an option of several simulator commands that operate within the Questa SIM GUI environment. The expressions help you locate and examine objects within the

List and Wave windows (expressions may also be used through the **Edit > Search** menu in both windows). The commands that use the expression format are:

compare add
compare clock
compare configure
configure
examine
searchlog
virtual function
virtual signal
down, left, right, up.

Expression Typing

GUI expressions are typed. The supported types consist of the following scalar and array types.

Scalar Types

The scalar types are as follows: boolean, integer, real, time (64-bit integer), enumeration, and signal state. Signal states are represented by the nine VHDL std_logic states: ’U’ ’X’ ’0’ ’1’ ’Z’ ’W’ ’L’ ’H’ and ’-’.

Verilog states 0, 1, x, and z are mapped into these states and the Verilog strengths are ignored. Conversion is done automatically when referencing Verilog nets or registers.

SystemC scalar types supported are: all the C/C++ types except class, structure, union, and array, as well as SystemC types sc_logic and sc_bit.

Array Types

The supported array types are signed and unsigned arrays of signal states. This would correspond to the VHDL std_logic_array type. Verilog registers are automatically converted to these array types. The array type can be treated as either UNSIGNED or SIGNED, as in the IEEE std_logic_arith package. Normally, referencing a signal array causes it to be treated as UNSIGNED by the expression evaluator; to cause it to be treated as SIGNED, use casting as described below. Numeric operations supported on arrays are performed by the expression evaluator via Questa SIM’s built-in numeric_standard (and similar) package routines. The expression evaluator selects the appropriate numeric routine based on SIGNED or UNSIGNED properties of the array arguments and the result.

The enumeration types supported are any VHDL enumerated type. Enumeration literals may be used in the expression as long as some variable of that enumeration type is referenced in the expression. This is useful for sub-expressions of the form:

```
(/memory/state == reading)
```

The supported SystemC aggregate types are the C/C++ array types: union, class, structure, and array. Also supported are the SystemC array types: sc_bv<w>, sc_lv<w>, sc_int<w>, etc.

Expression Syntax

GUI expressions generally follow C-language syntax, with both VHDL-specific and Verilog-specific conventions supported. These expressions are not parsed by the Tcl parser, and so do not support general Tcl; parentheses should be used rather than braces. Procedure calls are not supported.

A GUI expression can include the following elements: Tcl macros, constants, array constants, variables, array variables, signal attributes, operators, and casting.

Tcl Macros

Macros are useful for pre-defined constants or for entire expressions that have been previously saved. The substitution is done only once, when the expression is first parsed.

Macro syntax is:

```
$<name>
```

Substitutes the string value of the Tcl global variable <name>.

Constants

Table 1-10. Constants Supported for GUI Expressions

Type	Values
boolean value	true false TRUE FALSE
integer	[0-9]+
real number	<int>|([<int>].<int>[exp]) where the optional [exp] is: (e|E)[+|-][0-9]+
time	integer or real optionally followed by time unit
enumeration	VHDL user-defined enumeration literal
single bit constants	expressed as any of the following: 0 1 x X z Z U H L W ’U’ ’X’ ’0’ ’1’ ’Z’ ’W’ ’L’ ’H’ ’-’ 1’b0 1’b1

Array Constants, Expressed in Any of the Following Formats

Table 1-11. Array Constants Supported for GUI Expressions

Type	Values
VHDL # notation	<int>#<alphanum>[#] Example: 16#abc123#
VHDL bitstring	"(U|X|0|1|Z|W|L|H|-)*" Example: "11010X11"
Verilog notation	[-][<int>]’(b|B|o|O|d|D|h|H) <alphanum> (where <alphanum> includes 0-9, a-f, A-F, and ’-’) Example: 12’hc91 (This is the preferred notation because it removes the ambiguity about the number of bits.)
Based notation	0x…, 0X…, 0o…, 0O…, 0b…, OB… Questa SIM automatically zero fills unspecified upper bits.

Variables

Table 1-12. Variables Supported for GUI Expressions

Variable	Type
Name of a signal	The name may be a simple name, a VHDL or Verilog style extended identifier, or a VHDL or Verilog style path. The signal must be one of the following types: -- VHDL signal of type INTEGER, REAL, or TIME -- VHDL signal of type std_logic or bit -- VHDL signal of type user-defined enumeration -- Verilog net, Verilog register, Verilog integer, or Verilog real -- SystemC primitive channels of type scalar (e.g. bool, int, etc.)

Table 1-12. Variables Supported for GUI Expressions (cont.)

Variable	Type
NOW	Returns the value of time at the current location in the WLF file as the WLF file is being scanned (not the most recent simulation time).

Array variables

Table 1-13. Array Variables Supported for GUI Expressions

Variable	Type
Name of a signal	-- VHDL signals of type bit_vector or std_logic_vector -- Verilog register -- Verilog net array -- SystemC primitive channels of type vector (e.g. sc_bv, sc_int, etc.) A subrange or index may be specified in either VHDL or Verilog syntax. Examples: mysignal(1 to 5), mysignal[1:5], mysignal (4), mysignal [4]

Signal attributes

```
<name>'event
<name>'rising
<name>'falling
<name>'delayed()
<name>'hasX
```

The 'delayed attribute lets you assign a delay to a VHDL signal. To assign a delay to a signal in Verilog, use "#" notation in a sub-expression (e.g., *#-10 /top/signalA*).

The hasX attribute lets you search for signals, nets, or registers that contains an X (unknown) value.

See Examples of Expression Syntax below for further details on 'delayed and 'hasX.

Operators

Table 1-14. Operators Supported for GUI Expressions

Operator	Description	Kind
+	arithmetic add	arithmetic
/	arithmetic divide	arithmetic
mod/MOD	arithmetic modulus	arithmetic
*	arithmetic multiply	arithmetic
rem/REM	arithmetic remainder	arithmetic
-	arithmetic subtract	arithmetic
&	concat	arithmetic
<name>'delayed(<time>)	delayed signal (<time>)	attributes
<name>'falling	Falling edge	attributes
<name>'rising	Rising edge	attributes
<name>'event	Value change	attributes
<name>'hasX	Value has an X	attributes
and, AND	bitwise and	bitwise logical
nand, NAND	bitwise nand	bitwise logical
nor, NOR	bitwise nor	bitwise logical
or, OR	bitwise or	bitwise logical
xnor, XNOR	bitwise xnor	bitwise logical
xor, XOR	bitwise xor	bitwise logical
rol, ROL	rotate left	bitwise logical
ror, ROR	rotate right	bitwise logical
sla, SLA	shift left arithmetic	bitwise logical
sll, SLL	shift left logical	bitwise logical
sra, SRA	shift right arithmetic	bitwise logical
srl, SRL	shift right logical	bitwise logical
not, NOT, ~	unary bitwise inversion	bitwise logical
&&	boolean and	boolean
!	boolean not	boolean
||	boolean or	boolean

Table 1-14. Operators Supported for GUI Expressions (cont.)

Operator	Description	Kind
==	equal	boolean
===	exact equal[1]	boolean
!==	exact not equal[1]	boolean
>	greater than	boolean
>=	greater than or equal	boolean
<	less than	boolean
<=	less than or equal	boolean
!=, /=	not equal	boolean
&<vector_expr>	AND reduction	reduction
|<vector_expr>	OR reduction	reduction
^<vector_expr>	XOR reduction	reduction

1. This operator is allowed to be compatible with other simulators.

Table 1-15. Precedence of GUI Expression Operators

Operator	Kind
delayed(), 'falling, 'rising, 'event, 'hasX	attributes
&, |, ^	unary
!, not, NOT, ~	boolean
/, mod, MOD, *, rem, REM	arithmetic
nand, NAND, nor, NOR	bitwise logical
and, AND	bitwise logical
xor, XOR, xnor, XNOR	bitwise logical
or, OR	bitwise logical
+, -	arithmetic
&	concat
rol, ROL, ror, ROR, sla, SLA, sll, SLL, sra, SRA, srl, SRL	bitwise logical
>, >=, <, <=	boolean
==, ===, !==, !=, /=	boolean
&&	boolean

Table 1-15. Precedence of GUI Expression Operators (cont.)

Operator	Kind
||	boolean

Note

Arithmetic operators use the std_logic_arith package.

Casting

Table 1-16. Casting Conversions Supported for GUI Expressions

Casting	Description
(bool)	convert to boolean
(boolean)	convert to boolean
(int)	convert to integer
(integer)	convert to integer
(real)	convert to real
(time)	convert to 64-bit integer
(std_logic)	convert to 9-state signal value
(signed)	convert to signed vector
(unsigned)	convert to unsigned vector
(std_logic_vector)	convert to unsigned vector

Examples of Expression Syntax

```
/top/bus & $bit_mask
```

This expression takes the bitwise AND function of signal */top/bus* and the array constant contained in the global Tcl variable bit_mask.

```
clk'event && (/top/xyz == 16'hffae)
```

This expression evaluates to a boolean true when signal *clk* changes and signal */top/xyz* is equal to hex ffae; otherwise is false.

```
clk'rising && (mystate == reading) && (/top/u3/addr == 32'habcd1234)
```

Evaluates to a boolean true when signal *clk* just changed from low to high and signal *mystate* is the enumeration reading and signal */top/u3/addr* is equal to the specified 32-bit hex constant; otherwise is false.

```
(/top/u3/addr and 32'hff000000) == 32'hac000000
```

Evaluates to a boolean true when the upper 8 bits of the 32-bit signal */top/u3/addr* equals hex ac.

```
/top/signalA'delayed(10ns)
```

This expression returns */top/signalA* delayed by 10 ns.

```
/top/signalA'delayed(10 ns) && /top/signalB
```

This expression takes the logical AND of a delayed */top/signalA* with */top/signalB*.

```
virtual function { (#-10 /top/signalA) && /top/signalB}
mySignalB_AND_DelayedSignalA
```

This evaluates */top/signalA* at 10 simulation time steps before the current time, and takes the logical AND of the result with the current value of */top/signalB*. The '#' notation uses positive numbers for looking into the future, and negative numbers for delay. This notation does not support the use of time units.

```
((NOW > 23 us) && (NOW < 54 us)) && clk'rising && (mode == writing)
```

Evaluates to a boolean true when WLF file time is between 23 and 54 microseconds, *clk* just changed from low to high, and signal mode is enumeration writing.

```
searchlog -expr {dbus'hasX} {0 ns} dbus
```

Searches for an 'X' in *dbus*. This is equivalent to the expression: *{dbus(0) == 'x' || dbus(1) == 'x'}* . . .. This makes it possible to search for X values without having to write a type specific literal.

Signal and Subelement Naming Conventions

Questa SIM supports naming conventions for VHDL and Verilog signal pathnames, VHDL array indexing, Verilog bit selection, VHDL subrange specification, and Verilog part selection.

All supported naming conventions for VHDL and Verilog are valid for SystemC designs.

Examples in Verilog and VHDL syntax:

```
top.chip.vlogsig
/top/chip/vhdlsig
vlogsig[3]
vhdlsig(9)
vlogsig[5:2]
vhdlsig(5 downto 2)
```

Grouping and Precedence

Operator precedence generally follows that of the C language, but we recommend liberal use of parentheses.

Concatenation of Signals or Subelements

Elements in the concatenation that are arrays are expanded so that each element in the array becomes a top-level element of the concatenation. But for elements in the concatenation that are records, the entire record becomes one top-level element in the result.

To specify that the records be broken down so that their subelements become top-level elements in the concatenation, use the **concat_flatten** directive. Currently we do not support leaving full arrays as elements in the result. (Please let us know if you need that option.)

If the elements being concatenated are of incompatible base types, a VHDL-style record will be created. The record object can be expanded in the Objects and Wave windows just like an array of compatible type elements.

Concatenation Syntax for VHDL

```
<signalOrSliceName1> & <signalOrSliceName2> & ...
```

Concatenation Syntax for Verilog

```
&{<signalOrSliceName1>, <signalOrSliceName2>, ... }
&{<count>{<signalOrSliceName1>}, <signalOrSliceName2>, ... }
```

Note that the concatenation syntax begins with "&{" rather than just "{". Repetition multipliers are supported, as illustrated in the second line. The repetition element itself may be an arbitrary concatenation subexpression.

Concatenation Directives

A concatenation directive (as illustrated below) can be used to constrain the resulting array range of a concatenation or influence how compound objects are treated. By default, the concatenation will be created with a descending index range from (*n*-1) down to 0, where *n* is the number of elements in the array.

```
(concat_range 31:0)<concatenationExpr> # Verilog syntax
(concat_range (31:0))<concatenationExpr> # Also Verilog syntax
(concat_range (31 downto 0))<concatenationExpr> # VHDL syntax
```

The **concat_range** directive completely specifies the index range.

```
(concat_ascending) <concatenationExpr>
```

The **concat_ascending** directive specifies that the index start at zero and increment upwards.

```
(concat_flatten) <concatenationExpr>
```

The **concat_flatten** directive flattens the signal structure hierarchy.

```
(concat_noflatten) <concatenationExpr>
```

The concat_noflatten directive groups signals together without merging them into one big array. The signals become elements of a record and retain their original names. When expanded, the new signal looks just like a group of signals. The directive can be used hierarchically with no limits on depth.

```
(concat_sort_wild_ascending) <concatenationExpr>
```

The **concat_sort_wild_ascending** directive gathers signals by name in ascending order (the default is descending).

```
(concat_reverse) <concatenationExpr>
```

The **concat_reverse** directive reverses the bits of the concatenated signals.

Examples of Concatenation

```
&{ "mybusbasename*" }
```

Gathers all signals in the current context whose names begin with "mybusbasename", sorts those names in descending order, and creates a bus with index range (*n*-1) downto 0, where *n* is the number of matching signals found. (Note that it currently does not derive the index name from the tail of the one-bit signal name.)

```
(concat_range 13:4)&{ "mybusbasename*" }
```

Specifies the index range to be 13 downto 4, with the signals gathered by name in descending order.

```
(concat_ascending)&{ "mybusbasename*" }
```

Specifies an ascending range of 0 to *n*-1, with the signals gathered by name in descending order.

```
(concat_ascending)((concat_sort_wild_ascending)&{"mybusbasename*" })
```

Specifies an ascending range of 0 to *n*-1, with the signals gathered by name in ascending order.

```
(concat_reverse)(bus1 & bus2)
```

Specifies that the bits of bus1 and bus2 be reversed in the output virtual signal.

Record Field Members

Arbitrarily-nested arrays and records are supported, but operators will only operate on one field at a time. That is, the expression *{a == b}* where *a* and *b* are records with multiple fields, is not supported.

This would have to be expressed as:

```
{(a.f1 == b.f1) && (a.f2 == b.f2) ...}
```

Examples:

```
vhdlsig.field1
vhdlsig.field1.subfield1
vhdlsig.(5).field3
vhdlsig.field4(3 downto 0)
```

Searching for Binary Signal Values in the GUI

When you use the GUI to search for signal values displayed in 4-state binary radix, you should be aware of how Questa SIM maps between binary radix and std_logic. The issue arises because there is no “un-initialized” value in binary, while there is in std_logic. So, Questa SIM relies on mapping tables to determine whether a match occurs between the displayed binary signal value and the underlying std_logic value.

This matching algorithm applies only to searching using the GUI. It does not apply to VHDL or Verilog test benches.

For comparing VHDL std_logic/std_ulogic objects, Questa SIM uses the table shown below. An entry of “0” in the table is “no match”; an entry of “1” is a “match”; an entry of “2” is a match only if you set the Tcl variable **STDLOGIC_X_MatchesAnything** to 1. Note that *X* will match a *U*, and - will match anything.

Table 1-17. VHDL Logic Values Used in GUI Search

Search Entry	**Matches as follows:**								
	U	**X**	**0**	**1**	**Z**	**W**	**L**	**H**	**-**
U	1	1	0	0	0	0	0	0	1
X	1	1	2	2	2	2	2	2	1
0	0	2	1	0	0	0	1	0	1
1	0	2	0	1	0	0	0	1	1
Z	0	2	0	0	1	0	0	0	1
W	0	2	0	0	0	1	0	0	1
L	0	2	1	0	0	0	1	0	1
H	0	2	0	1	0	0	0	1	1
-	1	1	1	1	1	1	1	1	1

For comparing Verilog net values, Questa SIM uses the table shown below. An entry of “2” is a match only if you set the Tcl variable “VLOG_X_MatchesAnything” to 1.

Table 1-18. Verilog Logic Values Used in GUI Search

Search Entry	Matches as follows:			
	0	**1**	**Z**	**X**
0	1	0	0	2
1	0	1	0	2
Z	0	0	1	2
X	2	2	2	1

This table also applies to SystemC types: sc_bit, sc_bv, sc_logic, sc_int, sc_uint, sc_bigint, sc_biguint.

Chapter 2
Commands

This chapter describes Questa SIM commands that you can enter either on the command line of the Main window or in a DO file. Some commands are automatically entered on the command line when you use the graphical user interface.

Note that, in addition to the simulation commands listed in this chapter, you can also use the Tcl commands described in the Tcl man pages (use the Main window menu selection: **Help > Tcl Man Pages**).

Table 2-1 provides a brief description of each Questa SIM command and whether the command is supported for use in batch simulation mode (vsim **-batch**), and/or command-line mode (vsim **-c**). Refer to "Modes of Operation" for more information about batch and command-line simulation. For more information on command details, arguments, and examples, click the link in the Command name column.

Table 2-1. Supported Commands

Command name	Action	-batch	-c
.main clear	clears the Main window transcript	N	N
abort	halts the execution of a DO file interrupted by a breakpoint or error	Y	Y
add atv	opens an Assertion Thread View (ATV) window for the specified assert or cover directive (designated by its pathname), at the specified evaluation attempt start time	N	N
add button	adds a user-defined button to the Main window button bar	N	N
add dataflow	adds the specified object(s) to the Dataflow window	N	N
add list	lists VHDL signals and variables, and Verilog nets and registers, and their values in the List window	Y	Y
add log	also known as the **log** command; see log	Y	Y
add memory	opens the specified memory in the MDI frame of the Main window	N	N
add message	used within a DO file or script and specifies a user defined runtime message that is sent to the transcript and *.wlf* files	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
add schematic	adds the specified object(s) to the Schematic window	N	N
add testbrowser	adds .ucdb files to the Test Management Browser	N	N
add watch	adds signals or variables to the Watch window	N	N
add wave	adds VHDL signals and variables, and Verilog nets and registers to the Wave window	Y	Y
add_cmdhelp	adds an entry to the command-line help; use the help command to display the help text	N	Y
add_menu	adds a menu to the menu bar of the specified window, using the specified menu name	N	N
add_menucb	creates a checkbox within the specified menu of the specified window	N	N
add_menuitem	creates a menu item within the specified menu of the specified window	N	N
add_separator	adds a separator as the next item in the specified menu path in the specified window	N	N
add_submenu	creates a cascading submenu within the specified menu path of the specified window	N	N
alias	creates a new Tcl procedure that evaluates the specified commands	Y	N
archive load	allows you to load an archived debug database (.dbar) file that was previously created with the archive write command	N	N
archive write	allows you to create a debug archive file, with the file extension .dbar, that contains one or more WLF files, debug information captured from the design library, an optional connectivity debug database file, and optional HDL source files	N	N
assertion action	allows you to set the assertion action for assertion starts, fails, passes, or antecedent matches	Y	Y
assertion active	instructs the simulator to report on any active assertion directives at the end of simulation (EOS)	Y	Y
assertion count	returns the sum of the assertion failure counts for the specified set of assertion directive instances	Y	Y
assertion enable	enables and disables assertions and cover directives	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
assertion fail	configures fail tracking for SystemVerilog and PSL assertions	Y	Y
assertion pass	configures pass tracking for SystemVerilog and PSL assertions	Y	Y
assertion profile	generates a fine grained profile of memory usage for assertions and cover directives	Y	Y
assertion report	returns a status report for each SystemVerilog or PSL assertion matching the path specification	Y	Y
atv log	enables or disables assertion thread viewing (ATV) for the specified assertion(s)	Y	Y
batch_mode	returns a 1 if Questa SIM is operating in batch mode, otherwise returns a 0	Y	Y
bd	deletes a breakpoint	Y	Y
bookmark add wave	adds a bookmark to the specified Wave window	N	N
bookmark delete wave	deletes bookmarks from the specified Wave window	N	N
bookmark goto wave	zooms and scrolls a Wave window using the specified bookmark	N	N
bookmark list wave	displays a list of available bookmarks	N	N
bp	sets a breakpoint	Y	Y
call	calls SystemVerilog static, package, and class functions	Y	Y
cd	changes the Questa SIM local directory to the specified directory	Y	Y
cdbg	provides command-line equivalents of the menu options that are available for C Debug	N	Y
change	modifies the value of a VHDL variable or Verilog register variable	Y	Y
change_menu_cmd	changes the command to be executed for a specified menu item label, in the specified menu, in the specified window	N	N
check contention add	enables contention checking for the specified nodes	Y	Y
check contention config	writes checking messages to a file	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
check contention off	disables contention checking for the specified nodes	Y	Y
check float add	enables float checking for the specified nodes	Y	Y
check float config	writes checking messages to a file	Y	Y
check float off	disables float checking for the specified nodes	Y	Y
check stable off	disables stability checking	Y	Y
check stable on	enables stability checking on the entire design	Y	Y
checkpoint	saves the state of your simulation	Y	Y
classinfo ancestry	returns SystmVerilog class inheritance hierarchy for the specified class type	Y	Y
classinfo descriptive	returns the descriptive class name for the specified authoritative class name	Y	Y
classinfo find	reports on the current state for a specified class instance	Y	Y
classinfo implements	displays a list of which classes implement interface classes	Y	Y
classinfo instances	displays a list of class items of a specified class	Y	Y
classinfo interfaces	displays a list of interface classes or information about interface classes	Y	Y
classinfo isa	returns a list of all classes extended from the specified class type	Y	Y
classinfo report	displays a detailed report of a set of classes and their usage	Y	Y
classinfo stats	displays statistics about the number of class types and instances	Y	Y
classinfo trace	displays active references to a specified class instance	Y	Y
classinfo types	displays a list of classes or information about class types	Y	Y
compare add	compares signals in a reference design against signals in a test design	N	Y
compare annotate	marks a compare difference as "ignore" or tags it with a text message	N	Y
compare clock	defines a clock to be used with clocked-mode comparisons	N	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
compare configure	modifies options for compare signals or regions	N	Y
compare continue	continues difference computation that had been suspended	N	Y
compare delete	deletes a signal or region from the current comparison	N	Y
compare end	closes the currently open comparison	N	Y
compare info	lists the results of the comparison	N	Y
compare list	lists all the compare add commands currently in effect	N	Y
compare options	sets defaults for options used in other compare commands	N	Y
compare reload	reloads a comparison previously saved with the compare savediffs command	N	Y
compare reset	clears the current compare differences	N	Y
compare run	runs the comparison on selected signals	N	Y
compare savediffs	saves comparison differences to a file that can be reloaded later	N	Y
compare saverules	saves comparison setup information to a file that can be reloaded later	N	Y
compare see	displays a comparison difference in the Wave window	N	Y
compare start	starts a new dataset comparison	N	Y
compare stop	halts active difference computation	N	Y
compare update	updates the comparison differences	N	Y
configure	invokes the List or Wave widget configure command for the current default List or Wave window	N	Y
context	provides several operations on a context's name	Y	Y
coverage analyze	displays test oriented information based on a vcover merge operation	N	Y
coverage attribute	displays attributes in the currently loaded database	Y	Y
coverage clear	clears all coverage data obtained during previous run commands. Undocumented for 6.4	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
coverage create	creates covergroups, coverpoints, crosses and bins under a specified parent in an existing UCDB file	N	Y
coverage edit	opens a coverage dataset (.ucdb) to edit the contents. Used only in Coverage View mode (vsim -viewcov)	N	Y
coverage exclude	loads an exclusion filter file; or, allows you to exclude specific objects, files, design units, lines in a source file or rows within a table, and so on	Y	Y
coverage goal	sets the value of UCDB-wide goals	N	Y
coverage loadtestassoc	loads test vector data into an open UCDB. This command is necessary in order to use any test-associated features or view any test-associated data when in Coverage View mode (vsim -viewcov).	Y	Y
coverage open	opens UCDB datasets for viewing in the GUI in Coverage View mode	N	Y
coverage ranktest	ranks coverage data according to user-specified tests	N	Y
coverage report	produces a textual output of the coverage statistics that have been gathered up to this point	Y	Y
coverage save	saves current coverage statistics to a file that can be reloaded later, preserving instance-specific information	Y	Y
coverage tag	adds or removes tags from specified objects	N	Y
coverage testnames	displays test names in the current UCDB file loaded	N	Y
coverage unlinked	reports on testplan item with no coverage associated, or design coverage not associated with a testplan item	N	Y
coverage weight	sets a global per-type weight for total coverage calculations	N	Y
dataset alias	assigns an additional name to a dataset	N	Y
dataset clear	clears the current simulation WLF file	N	Y
dataset close	closes a dataset	N	Y
dataset config	configures WLF file settings after dataset is open	N	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
dataset current	opens the specified dataset and sets the GUI context to the last selected context of the specified dataset	N	Y
dataset info	reports information about the specified dataset	N	Y
dataset list	lists the open dataset(s)	N	Y
dataset open	opens a dataset and references it by a logical name	N	Y
dataset rename	changes the logical name of an opened dataset	N	Y
dataset restart	unloads specified or current dataset	N	Y
dataset save	saves data from the current WLF file to a specified file	N	Y
dataset snapshot	saves data from the current WLF file at a specified interval	N	Y
delete	removes objects from either the List or Wave window	N	Y
describe	displays information about the specified HDL object	Y	Y
disablebp	turns off breakpoints and when commands	Y	Y
disable_menu	disables the specified menu within the specified window	N	N
disable_menuitem	disables the specified menu item within the specified menu path of the specified window	N	N
do	executes commands contained in a DO file	Y	Y
down	searches for signal transitions or values in the specified List window	N	Y
drivers	displays in the Main window the current value and scheduled future values for all the drivers of a specified VHDL signal or Verilog net	Y	Y
dumplog64	dumps the contents of the *vsim.wlf* file in a readable format		
echo	displays a specified message in the Main window	Y	Y
edit	invokes the editor specified by the EDITOR environment variable	N	N
enablebp	turns on breakpoints and when commands turned off by the disablebp command	Y	Y
enable_menu	enables a previously-disabled menu	N	N

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
enable_menuitem	enables a previously-disabled menu item	N	N
encoding	translates between Unicode Tcl strings and a named encoding	Y	Y
environment	displays or changes the current dataset and region environment	Y	Y
examine	examines one or more objects, and displays current values (or the values at a specified previous time) in the Main window	Y	Y
exit	exits the simulator and the Questa SIM application	Y	Y
fcover configure	enables, disables, and sets coverage targets for PSL coverage directives	Y	Y
find	displays the full pathnames of all objects in the design whose names match the name specification you provide	N	Y
find connections	returns the set of nets that are electrically equivalent to a specified net. It is only available during a live simulation	N	Y
find drivers	traces backward in time to find the active driver(s), processes, or first elements of a specified signal or signal event	N	Y
find infiles	searches the specified files and prints to the Transcript window those lines from the files that match the specified pattern	Y	Y
find insource	searches all source files related to the current design and prints to the Transcript window those lines from the files that match the specified pattern	Y	Y
force	applies stimulus to VHDL signals and Verilog nets	Y	Y
formatTime	global format control for all time values displayed in the GUI	Y	Y
fsm list	returns information about the recognized finite state machines	N	N
fsm properties	returns information about a specified finite state machine	N	N
fsm view	displays a specified finite state machine in the FSM List window	N	N
gc configure	configures when the SystemVerilog Garbage Collector will run	N	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
gc run	runs the SystemVerilog Garbage Collector	N	Y
gdb dir	sets the source directory for FLI/PLI/VPI C source code when using C Debug	N	Y
getactivecursortime	gets the time of the active cursor in the Wave window	N	N
getactivemarkertime	gets the time of the active marker in the List window	N	N
help	displays in the Main window a brief description and syntax for the specified command	N	Y
history	lists the commands executed during the current session	Y	Y
jobspy	controls and monitors batch jobs	N	Y
layout	allows you to perform operations on GUI layouts	N	N
lecho	takes one or more Tcl lists as arguments and pretty-prints them to the Main window	Y	Y
left	searches left (previous) for signal transitions or values in the specified Wave window	N	N
log	creates a wave log format (WLF) file containing simulation data for all objects whose names match the provided specifications	Y	Y
lshift	takes a Tcl list as an argument and shifts it in-place one place to the left, eliminating the left-most element	Y	Y
lsublist	returns a sublist of the specified Tcl list that matches the specified Tcl glob pattern	Y	Y
mem compare	compares the selected memory to a reference memory or file	Y	Y
mem display	displays the memory contents of a selected instance to the screen	Y	Y
mem list	displays a flattened list of all memory instances in the current or specified context after a design has been elaborated	Y	Y
mem load	updates the simulation memory contents of a specified instance	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
mem save	saves the contents of a memory instance to a file in any of the supported formats: Verilog binary, Verilog hex, and MTI memory pattern data	Y	Y
mem search	finds and prints to the screen the first occurring match of a specified memory pattern in the specified memory instance	Y	Y
next	continues a search; see the search command	N	N
noforce	removes the effect of any active force commands on the selected object	Y	Y
nolog	suspends writing of data to the WLF file for the specified signals	Y	Y
notepad	opens a simple text editor	N	N
noview	closes a window or set of windows in the Questa SIM GUI	N	N
nowhen	deactivates selected when commands	Y	Y
onbreak	specifies command(s) to be executed when running a DO file script that encounters a breakpoint in the source code; in effect only during a run command	Y	Y
onElabError	specifies one or more commands to be executed when an error is encountered during elaboration; in effect only during a vsim command	Y	Y
onerror	specifies one or more commands to be executed when a Tcl command in a dofile encounters an error; not dependent on a run command	Y	Y
onfinish	controls simulator behavior when encountering $finish or sc_stop() in the design code	Y	Y
pa autotestplan	creates a test plan file based on coverage information in your power aware simulation	Y	Y
pa msg	controls display of warning and error message during Power Aware dynamic checks	Y	Y
pa report	This command is the final step of the power aware report generation flow, which writes out all of the power aware reports to the current working directory	Y	Y
pause	interrupts the execution of a DO file script	Y	Y
pop	moves one level up the C callstack	N	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
power add	specifies the signals or nets to track for power information	Y	Y
power off	works in conjunction with the power add command to make vsim stop updating toggle activity data for the specified signal or net	Y	Y
power on	works in conjunction with the power add command to make vsim begin or resume updating toggle activity data for the specified signal or net	Y	Y
power report	writes out the power information for the specified signals or nets	Y	Y
power reset	resets power information to zero for the signals or nets specified with the power add command	Y	Y
precision	determines how real numbers display in the GUI	Y	Y
printenv	echoes to the Main window the current names and values of all environment variables	Y	Y
process report	creates textual report of all processes displayed in the Process window	Y	Y
profile clear	clears any statistical performance or memory allocation data that has been gathered during previous **run** commands	Y	Y
profile interval	selects the frequency with which the profiler collects samples during a run command	Y	Y
profile off	disables runtime statistical performance and memory allocation profiling	Y	Y
profile on	enables runtime profiling of where your simulation is spending its time and where memory is allocated	Y	Y
profile open	opens a database containing profile data created by profile save	Y	Y
profile option	allows various profiling options to be changed	Y	Y
profile reload	reads in raw profile data from an external file created during memory allocation profiling	Y	Y
profile report	produces a textual output of the profiling statistics that have been gathered up to this point	Y	Y
profile save	aves profile data to an external database	Y	Y
profile summary	outputs profiling data that is broken into categories based on context in which the callstacks are found	N	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
project	performs common operations on new projects	N	Y
property list	changes one or more properties of the specified signal, net, or register in the List Window	N	N
property wave	changes one or more properties of the specified signal, net, or register in the Wave Window	N	N
push	moves one level down the C callstack	N	Y
pwd	displays the current directory path in the Main window	Y	Y
questasim	starts the QuestaSim GUI without prompting you to load a design; valid only for Windows platforms	Y	Y
quietly	turns off transcript echoing for the specified command	Y	Y
quit	exits the simulator	Y	Y
qverilog	compiles, optimizes, and simulates a Verilog or SystemVerilog design in one step	N	Y
radix	specifies the default radix to be used	Y	Y
radix define	creates or modifies a user-defined radix	Y	Y
radix delete	removes the radix definition from the named radix	Y	Y
radix list	returns the complete definition of a radix	Y	Y
radix names	returns a list of currently defined radix names	Y	Y
radix signal	sets or inspects radix values for the specified signal in the Objects, Locals, and Wave windows	Y	Y
readers	displays the names of all readers of the specified object	N	Y
report	displays the value of all simulator control variables, or the value of any simulator state variables relevant to the current simulation	Y	Y
restart	reloads the current dataset if the current dataset is not the active simulation ("sim") and resets the simulation time to zero, in effect acting just like a restart of a simulation	Y	Y
restore	restores the state of a simulation that was saved with a checkpoint command during the current invocation of vsim	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
resume	continues execution of a DO file script after a pause command or a breakpoint	Y	Y
right	searches right (next) for signal transitions or values in the specified Wave window	N	N
run	advances the simulation by the specified number of timesteps	Y	Y
runStatus	returns the current state of your simulation after issuing a run or step command	Y	Y
sccom	compiles SystemC design units	Y	Y
scgenmod	creates the equivalent SystemC foreign module declaration for a VHDL entity or Verilog module, and writes it to standard output	Y	Y
sdfcom	compiles SDF files	Y	Y
search	searches the specified window for one or more objects matching the specified pattern(s)	N	N
searchlog	searches one or more of the currently open logfiles for a specified condition	N	Y
see	displays the specified number of source file lines around the current execution line	Y	Y
seetime	scrolls the List or Wave window to make the specified time visible	N	Y
setenv	sets an environment variable	Y	Y
shift	shifts DO file script parameter values down one place	Y	Y
show	lists objects and subregions visible from the current environment	Y	Y
simstats	reports performance-related statistics about active simulations	Y	Y
stack down	moves down the call stack	Y	Y
stack frame	selects the specified call frame	Y	Y
stack level	reports the current call frame number	Y	Y
stack tb	is an alias for the tb command	Y	Y
stack up	moves up the call stack	Y	Y
status	lists all currently interrupted DO file scripts	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
step	steps to the next HDL statement	Y	Y
stop	stops simulation in batch files; used with the when command	Y	Y
suppress	prevents the specified message(s) from displaying	Y	Y
tb	displays a stack trace for the current process in the Transcript window	Y	Y
tcheck_set	modifies reporting or X generation status of a timing check	Y	Y
tcheck_status	prints the current status of timing checks to the Transcript window	Y	Y
Time	performs various numerical comparisons, operations, and conversions on simulation time values	Y	Y
toggle add	enables collection of toggle statistics for the specified nodes	Y	Y
toggle disable	disables collection of toggle statistics for the specified nodes	Y	Y
toggle enable	re-enables collection of toggle statistics for the specified nodes	Y	Y
toggle report	displays to the Transcript window a list of all nodes that have not transitioned to both 0 and 1 at least once	Y	Y
toggle reset	resets the toggle counts to zero for the specified nodes	Y	Y
tr color	modifies the color of a specific transaction or stream of transactions in a wave window, or all wave windows	N	Y
tr order	controls which attributes are visible in a transaction and the order in which they appear	N	Y
tr uid	displays to the Transcript window a list of all active transactions and their IDs	N	Y
transcribe	displays a command in the Transcript window, then executes the command	N	Y
transcript	controls echoing of commands executed in a DO file; also works at top level in command line and batch mode	Y[1]	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
transcript file	sets or queries the pathname for the transcript file	Y	Y
transcript path	returns the full pathname to the current transcript file	Y	Y
transcript sizelimit	sets or queries the current value for the transcript fileSizeLimit value, saves file when limit is reached, and continues simulation	Y	Y
triage dbfile	creates database of test data and/or simulation messages from UCDB, WLF, and/or LOG files	N	N
triage dbinsert	places one or more messages in triage database (TDB)	N	N
triage dbrefresh	clears old triage database and re-creates database using saved history of previous triage dbfile and triage dbinsert commands	N	N
triage query	provides programmatic access to contents of triage database for debug and scripting purposes	N	N
triage report	reads in a triage database and prints out a text report after applying specified filters	N	N
triage view	This command loads a triage database (*.tdb*) into the Verification Results Analysis window	N	N
tssi2mti	converts a vector file in Technology Standard Events Format (TSSI) into a sequence of force and run commands	N	Y
typespec	queries class names and class relationships of SystemVerilog classes	N	Y
ui_VVMode	specifies behavior when encountering user interface registration calls used by verification packages, such as AVM or OVM	Y	Y
unsetenv	deletes an environment variable	Y	Y
up	searches for signal transitions or values in the specified List window	N	N
uvm call	calls a UVM function	Y	Y
uvm configtracing	enables and disables printing of configuration database reads and writes as they occur	Y	Y
uvm displayobjections	displays current objections for a specified object or uvm_top	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
uvm handle	print simulation handle for a specified class instance	Y	Y
uvm mapmode	sets path mapping mode to look for UVM style paths or simulator paths	Y	Y
uvm printconfig	print configuration information for a specified component	Y	Y
uvm printfactory	print global factory information	Y	Y
uvm printstreams	print a list of transaction streams for a specified object	Y	Y
uvm printtopology	prints the global testbench topology	Y	Y
uvm setverbosity	sets the global reporting verbosity	Y	Y
uvm simpath	maps the path of a UVM hierarchical component to a simulator /uvm_root context hierarchy string	Y	Y
uvm traceobjections	enables and disables objections tracing for a component instance	Y	Y
uvm uvmpath	maps the path of a component from simulator hierarchy to a UVM full name	Y	Y
vcd add	adds the specified objects to the VCD file	Y	Y
vcd checkpoint	dumps the current values of all VCD variables to the VCD file	Y	Y
vcd comment	inserts the specified comment in the VCD file	Y	Y
vcd dumpports	creates a VCD file that captures port driver data	Y	Y
vcd dumpportsall	creates a checkpoint in the VCD file that shows the current values of all selected ports	Y	Y
vcd dumpportsflush	flushes the VCD buffer to the VCD file	Y	Y
vcd dumpportslimit	specifies the maximum size of the VCD file	Y	Y
vcd dumpportsoff	turns off VCD dumping and records all dumped port values as x	Y	Y
vcd dumpportson	turns on VCD dumping and records the current values of all selected ports	Y	Y
vcd file	specifies the filename and state mapping for the VCD file created by a vcd add command	Y	Y
vcd files	specifies filenames and state mapping for the VCD files created by the vcd add command; supports multiple VCD files	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
vcd flush	flushes the contents of the VCD file buffer to the VCD file	Y	Y
vcd limit	specifies the maximum size of the VCD file	Y	Y
vcd off	turns off VCD dumping and records all VCD variable values as x	Y	Y
vcd on	turns on VCD dumping and records the current values of all VCD variables	Y	Y
vcd2wlf	translates VCD files into WLF files	Y	Y
vcom	compiles VHDL design units	Y	Y
vcover attribute	displays attributes in the currently loaded database	Y	Y
vcover diff	reports the coverage differences between two UCDBs	Y	Y
vcover dump	prints to stdout a textual description of contents of UCDB file	Y	Y
vcover history	allows you to acces the details on how a specific UCDB file was created, regardless of whether that UCDB was created by multiple commands over time, through scripts or regression runs	Y	Y
vcover merge	merges multiple code or functional coverage data files offline	Y	Y
vcover ranktest	ranks the specified input files according to their contribution to cumulative coverage	Y	Y
vcover report	reports on multiple code or functional coverage data files offline	Y	Y
vcover stats	produces summary statistics from multiple coverage data files	Y	Y
vcover testnames	displays test names in the current UCDB file loaded	Y	Y
vdel	deletes a design unit from a specified library	Y	Y
vdir	lists the contents of a design library	Y	Y
vencrypt	encrypts Verilog code contained within encryption envelopes	Y	Y
verror	prints a detailed description of a message number	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
vgencomp	writes the equivalent VHDL component declaration for a Verilog module to standard output	Y	Y
vhencrypt	encrypts VHDL code contained within encryption envelopes	Y	Y
view	opens a Questa SIM window and brings it to the front of the display	N	N
virtual count	counts the number of currently defined virtuals that were not read in using a DO file script	N	Y
virtual define	prints the definition of a virtual signal or function in the form of a command that can be used to re-create the object	N	Y
virtual delete	removes the matching virtuals	N	Y
virtual describe	prints a complete description of the data type of one or more virtual signals	N	Y
virtual expand	produces a list of all the non-virtual objects contained in the virtual signal(s)	N	Y
virtual function	creates a new signal that consists of logical operations on existing signals and simulation time	N	Y
virtual hide	causes the specified real or virtual signals to not be displayed in the Objects window	N	Y
virtual log	causes the sim-mode dependent signals of the specified virtual signals to be logged by the simulator	N	Y
virtual nohide	redisplays a virtual previously hidden with the virtual hide command	N	Y
virtual nolog	stops the logging of the specified virtual signals	N	Y
virtual region	creates a new user-defined design hierarchy region	N	Y
virtual save	saves the definitions of virtuals to a file	N	Y
virtual show	lists the full path names of all the virtuals explicitly defined	N	Y
virtual signal	creates a new signal that consists of concatenations of signals and subelements	N	Y
virtual type	creates a new enumerated type	N	Y
vlib	creates a design library	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
vlog	compiles Verilog design units and SystemVerilog extensions	Y	Y
vmake	creates a makefile that can be used to reconstruct the specified library	Y	Y
vmap	defines a mapping between a logical library name and a directory	Y	Y
vopt	produces an optimized version of your design	Y	Y
vsim	loads a new design into the simulator	Y	Y
vsim<info>	returns information about the current vsim executable	Y	Y
vsim_break	stop the current simulation before completion	Y	Y
vsource	specifies an alternative file to use for the current source file	Y	Y
wave	commands for manipulating cursors, for zooming, and for adjusting the wave display view in the Wave window	N	N
wave create	creates an editable waveform that can be used to create stimulus and drive simulation	N	N
wave edit	edits a created waveform	N	N
wave export	exports created waveforms to a stimulus file	N	N
wave import	imports an EVCD file previously created with a wave export command	N	N
wave modify	modifies the parameters of a created waveform	N	N
when	instructs Questa SIM to perform actions when the specified conditions are met	Y	Y
where	displays information about the system environment	Y	Y
wlf2log	translates a Questa SIM WLF file to a QuickSim II logfile	Y	Y
wlf2vcd	translates a Questa SIM WLF file to a VCD file	Y	Y
wlfman	outputs information about or a new WLF file from an existing WLF file	Y	Y
wlfrecover	attempts to repair an incomplete WLF file	Y	Y
write cell_report	creates a report of cell instances in the design that are optimized	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
write format	records the names and display options in a file of the objects currently being displayed in the List or Wave window	N	Y
write list	records the contents of the most recently opened or specified List window in a list output file	N	Y
write preferences	saves the current GUI preference settings to a Tcl preference file	N	Y
write report	prints a summary of the design being simulated	Y	Y
write timing	prints timing information about the specified instance	Y	Y
write transcript	writes the contents of the Main window transcript to the specified file	N	Y
write tssi	records the contents of the default or specified List window in a “TSSI format” file	Y	Y
write wave	records the contents of the most currently opened or specified Wave window in PostScript format	N	N
xml2ucdb	converts an XML file into a .ucdb file	Y	Y

1. transcript on | off only are supported.

.main clear

This command clears the Main window Transcript window.

The behavior is the same as selecting **Edit > Clear** when the Transcript window is active.

Syntax

.main clear

Arguments

None

Related Topics

- Main Window
- Transcript Window
- transcript
- transcript file

abort

This command halts the execution of a DO file interrupted by a breakpoint or error.

When DO files are nested, you may choose to abort the last DO file script only, abort a specified number of nesting levels, or abort all DO files. You can specify this command within a DO file to return early.

Syntax

abort [<n> | all]

Arguments

- <n>

  (optional) The number of nested DO file script levels to abort. Specified as an integer greater than 0, where the default value is 1.

- all

  (optional) Instructs the tool to abort all levels of nested DO files.

Related Topics

- onbreak
- onElabError
- onerror

add atv

This command opens an Assertion Thread View (ATV) window for the specified assert or cover directive (designated by its pathname), at the specified evaluation attempt start time.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

add atv <pathname> <time>

Arguments

- <pathname>

  (required) Specifies the path of an assert or cover directive. Must be specified as the first argument to the add atv command.

- <time>

  (required) Specifies the start time of an evaluation attempt of the assert or cover directive instance. Must be specified as the second argument to the add atv command.

Related Topics

- atv log
- Viewing Assertion Threads in the ATV Window

add button

This command adds a user-defined button to the Main window button bar. New buttons are added to the right side of the Standard toolbar.

Returns the path name of the button widget created. You may want to remember this path name, which is similar to:

```
# .dockbar.tbf0.standard.tb.button_49
```

in case you ever want to remove the button.

To remove a button you have previously added you can use the **destroy** Tcl command with the button's path name as an argument, for example:

destroy .dockbar.tbf0.standard.tb.button_49

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

add button <text> {**<cmd>[; ...]**} [Disable | NoDisable] [{<option> <value> ...}]

Arguments

- <text>

  (required) Specifies the label to appear on the face of the button. Must be specified as the first argument to the add button command.

- <cmd>[; ...]

  (required) Specifies the command(s) to be executed when the button is clicked. Multiple commands are specified by separating them with a semicolon (;). Must be specified as the second argument to the add button command.

  If your command contains any spaces or non-alphanumeric characters you must enclose the command in braces ({}).

  You can echo the command and display the return value in the Transcript window by prefixing the command with the transcribe command. The transcribe command will also echo the results to the Transcript window.

- Disable | NoDisable

  (optional) Specifies the appearance of the button.

  Disable — (default) The button is inactive and grayed-out during a run.

  NoDisable — The button is active and available during a run.

- <option> <value>

  (optional) Specifies Tk button option(s) that you want to apply to the button. Must be preceded by Disable or NoDisable.

  <option> — Any legal Tk button option.

<value> — Specifies the value for the Tk button option(s).

Multiple <option><value> Tk button options are entered as a space separated list. You must enclose your option/value pairs in braces ({}).

Note

To specify any option/value pairs, you must specify either Disable or NoDisable.

You can access the Tk documentation for button options by selecting **Help > Tcl Man Pages**, which displays HTML help. You can then select the links: **Tk commands,** then **buttons**.

You can use any properties belonging to Tk button widget. Useful options are foreground color (**-fg**), background color (**-bg**), width (**-width**), and relief (**-relief**).

For a complete list of available options, use the configure command addressed to the newly created option. For example:

```
.dockbar.tbf0.standard.tb.button_51 config
```

Examples

- Create a button labeled “pwd” that invokes the transcribe command with the **pwd** Tcl command, and echoes the command and its results to the Transcript window. The button remains active during a run.

```
add button pwd {transcribe pwd} NoDisable
```

- Create a button labeled “date” that echoes the system date to the Transcript window. The button is disabled during a run; its colors are: blue foreground, yellow background, and red active background.

```
add button date {transcribe exec date} Disable \
    {-fg blue -bg yellow -activebackground red}
```

- Create a “doit” button and underline the second character of the label, the "o" of "doit".

```
add button doit {run 1000 ns; echo did it} Disable {-underline 1}
```

- Change the command that the button executes to "run 10000" and the button’s background color to red; you must know the button’s path name that was returned after the initial creation of the button.

```
.dockbar.tbf0.standard.tb.button_13 config -command {run 10000} -bg red
```

Related Topics

- transcribe

add dataflow

This command adds the specified process, signal, net, or register to the Dataflow window. Wildcards are allowed.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

add dataflow <object> ... [-connect <source_net> <destination_net>]
{[-in] [-out] [-inout] | [-ports]} [-internal] [-nofilter] [-recursive]

Arguments

- <object> ...

  (required unless specifying -connect) Specifies a process, signal, net, or register to add to the Dataflow window. Wildcards are allowed. Multiple objects are specified as a space separated list, Refer to the section "Wildcard Characters" for wildcard usage as it pertains to the add commands. Must be specified as the first argument to the add dataflow command.

- -connect <source_net> <destination_net>

  (optional) Computes and displays in the Dataflow window all paths between two nets.

  <source_net> — The net that originates the path search.

  <destination_net> — The net that terminates the path search.

- -in

  (optional) Specifies to add ports of mode IN.

- -inout

  (optional) Specifies to add ports of mode INOUT.

- -out

  (optional) Specifies to add ports of mode OUT.

- -ports

  (optional) Specifies to add all ports. This switch has the same effect as specifying -in, -out, and -inout together.

- -internal

  (optional) Specifies to add internal (non-port) objects.

- -nofilter

  (optional) Specifies that the *WildcardFilter* Tcl preference variable be ignored when finding signals or nets.

  The *WildcardFilter* Tcl preference variable identifies types to ignore when matching objects with wildcard patterns.

- -recursive

  (optional) Specifies that the scope of the search is to descend recursively into subregions. If omitted, the search is limited to the selected region.

  You can specify -r as an alias to this switch.

Examples

- Add all objects in the design to the dataflow window.

  add dataflow -r /*

- Add all objects in the region to the dataflow window.

  add dataflow *

Related Topics

- Automatically Tracing All Paths Between Two Nets
- Dataflow Window
- Using the WildcardFilter Preference Variable

add list

This command adds the following objects and their values to the List window:

- VHDL signals and variables
- Verilog nets and registers
- User-defined buses
- SystemC primitive channels (signals)

If you do not specify a port mode, such as -in or -out, this command displays all objects in the selected region with names matching the object name specification.

Refer to "Wildcard Characters" for wildcard usage as it pertains to the add commands.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

add list {**<object> ...** | **<object_name> {sig ...}**} [-allowconstants] [-depth <level>]
[-filter <f> | -nofilter <f>] {[-in] [-inout] [-out] | [-ports]} [-internal] [-label <name>]
[-nodelta] [-optcells] [-<radix_type> | -radix <type>]
[-radixenumnumeric | -radixenumsymbolic] [-recursive] [-trigger | -notrigger]
[-width <integer>] [-window <wname>]

Arguments

- <object> ...

  (required when <object_name >{sig ...} is not specified.) Specifies the name of the object to be listed. Multiple objects are entered as a space separated list. Wildcards are allowed. Refer to the section "Wildcard Characters" for wildcard usage as it pertains to the add commands. Must be specified as the first argument to the add list command.

  Note that the WildcardFilter Tcl preference variable identifies types to ignore when matching objects with wildcard patterns.

  You can add variables as long as they are preceded by the process name. For example:

  ```
  add list myproc/int1
  ```

  You must specify the <object> argument as the first argument to the add list command.

- <object_name> {sig ...}

  (required when <object> is not specified) Creates a user-defined bus with the specified object name containing the specified signals (sig) concatenated within the user-defined bus. Arguments, must be enclosed in braces ({ }). Must be specified as the second argument to the add list command.

sig — A space-separated list of signals, enclosed in braces ({}), that are included in the user-defined bus. The signals may be either scalars or various sized arrays as long as they have the same element enumeration type.

For example:

```
add list {mybus {a b y}}
```

- -allowconstants

  (optional) *For use with wildcard searches.* Specifies that constants matching the wildcard search should be added to the List window.

  This command does not add constants by default because they do not change.

- -depth <level>

  (optional) Restricts a recursive search, as specified with -recursive, to a certain level of hierarchy.

  <level> — an integer greater than or equal to zero.

  For example, if you specify -depth 1, the command descends only one level in the hierarchy.

- -filter <f> | -nofilter <f>

  (optional) Allows a one-time modification of the WildcardFilter in the command invocation. The add list command can take as many [-filter <f>] and [-nofilter <f>] arguments as you would like to specify. Valid filters, <f>, are exactly the same set of words that can be applied to the WildcardFilter. The filter(s) used during a command starts with the WildcardFilter and then applies the user specified filters, if any. The -filter values are added to the filter, the -nofilter values are removed from the filter. They are applied in the order specified so conflicts are resolved with the last specified wins.

- -in

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode IN if they match the *object* specification.

- -inout

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode INOUT if they match the *object* specification.

- -out

  (optional) *For use with wildcard searches*. Specifies that the scope of the search is to include ports of mode OUT if they match the *object* specification.

- -ports

  (optional) For use with wildcard searches. Specifies that the scope of the search is to include all ports. This switch has the same effect as specifying -in, -out, and -inout together.

- -internal

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include internal objects (non-port objects) if they match the *object* specification. VHDL variables are not selected.

- -label <name>

  (optional) Specifies an alternative signal name to be displayed as a column heading in the listing.

  <name> — Specifies the label to be used at the top of the column. You must enclose <name> in braces ({}) if it includes any spaces.

  This alternative name is not valid in a force or examine command.

  However, you can use the alternate name when invoking the search command with <window_name> specified as list.

- -nodelta

  (optional) Specifies that the delta column not be displayed when adding signals to the List window. Identical to configure list -delta none.

- -optcells

  (optional) *For use with wildcard searches*. Allows Verilog optimized cell ports to be visible when using wildcards. By default Verilog optimized cell ports are not selected even if they match the specified wildcard pattern.

- -<radix_type>

  (optional) Specifies the radix type for the objects that follow in the command. Valid entries (or any unique abbreviations) are: binary, ascii, unsigned, decimal, octal, hex, sfixed, symbolic, ufixed, time, and default.

  If no radix is specified for an enumerated type, the default radix is used. You can change the default radix for the current simulation using the radix command. You can change the default radix permanently by editing the DefaultRadix variable in the *modelsim.ini* file.

  If you specify a radix for an array of a VHDL enumerated type, Questa SIM converts each signal value to 1, 0, Z, or X.

- -radix <type>

  (optional) Specifies a user-defined radix. The -radix <type> switch can be used in place of the -<radix_type> switch. For example, -radix hexadecimal is the same as -hex.

  <type> — binary, ascii, unsigned, decimal, octal, hex, sfixed, symbolic, time, ufixed, and default.

  Refer to the radix command for information about sfixed and ufixed radix types.

  This option overrides the global setting of the default radix (the DefaultRadix variable in the *modelsim.ini* file) for the current simulation only.

- -radixenumnumeric

  This option overrides the global setting of the default radix (the DefaultRadix variable in the *modelsim.ini* file).

- -radixenumsymbolic

  (optional) Reverses the action of -radixenumnumeric and sets the global setting of the default radix (the DefaultRadix variable in the *modelsim.ini* file) to symbolic.

- -recursive

  (optional) *For use with wildcard searches*. Specifies that the scope of the search is to descend recursively into subregions. If omitted, the search is limited to the selected region. You can use the -depth argument to specify how far down the hierarchy to descend. You can use "-r" as an alias to this switch.

- -trigger | -notrigger

  (optional) Specifies whether objects should be updated in the List window when the objects change value.

  -trigger — (default) Update objects in the List Window when their values change.

  -notrigger — Do not update objects in the List Window when their values change.

- -width <integer>

  (optional) Formats the column width. The maximum width, when not specifying this argument is 30,000 characters, which you can override with this switch.

  integer — A positive integer specifying the column width in characters.

- -window <wname>

  (optional) Adds objects to the specified List window (for example, list2).

  <wname> — The window to add objects to. Used to specify a particular window when multiple instances of that window type exist.

  This option selects an existing window, but does not create a new window. Use the view command with the -new option to create a new window.

Examples

- List all objects in the design.

  add list -r /*

- List all objects in the region.

  add list *

- List all input ports in the region.

  add list -in *

- Display a List window containing three columns headed *a*, *sig*, and *array_sig(9 to 23)*.

  add list a -label sig /top/lower/sig {array_sig(9 to 23)}

- List *clk*, *a*, *b*, *c*, and *d* only when *clk* changes.

  add list clk -notrigger a b c d

- Lists *clk*, *a*, *b*, *c*, and *d* every 100 ns.

  config list -strobeperiod {100 ns} -strobestart {0 ns} -usestrobe 1
  add list -notrigger clk a b c d

- Creates a user-defined bus named "mybus" consisting of three signals; the bus is displayed in hex.

  add list -hex {mybus {msb {opcode(8 downto 1)} data}}

- Lists the object *vec1* using symbolic values, lists *vec2* in hexadecimal, and lists *vec3* and *vec4* in decimal.

  add list vec1 -hex vec2 -dec vec3 vec4

- Open a new List window with "SV_Signals" as its title, then add signals to it.

  set SV_Signals [view list -new -title SV_Signals]

  add list -window $SV_Signals /top/mysignals

  The custom window title "SV_Signals" is saved as a TCL variable, then called using the '$' prefix.

Related Topics

- add wave
- log
- Extended Identifiers
- Using the WildcardFilter Preference Variable

add memory

This command displays the contents and sets the address and data radix of the specified memory in the MDI frame of the Main window.

Refer to "Wildcard Characters" for wildcard usage as it pertains to the add commands.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

add memory [-addressradix {decimal | hex}] [-dataradix <type>]
[-radixenumnumeric | -radixenumsymbolic] [-wordsperline <num>] **<object_name> ...**

Arguments

- -addressradix {decimal | hex}

  (optional) Specifies the address radix for the memory display.

  decimal — (default) Sets the radix to decimal. You can abbreviate this argument to "d".

  hex — Sets the radix to hexadecimal. You can abbreviate this argument to "h".

- -dataradix <type>

  (optional) Specifies the data radix for the memory display. If you do not specify this switch, the command uses the global default radix.

  <type> — binary, ascii, unsigned, decimal, octal, hex, sfixed, symbolic, time, ufixed, and default.

  Refer to the radix command for information about sfixed and ufixed radix types.

  If you do not specify a radix for an enumerated type, the command uses the symbolic representation.

  You can change the default radix for the current simulation using the radix command. You can change the default radix permanently by editing the DefaultRadix variable in the *modelsim.ini* file. Changing the default radix does not change the radix of the currently displayed memory. Use the add memory command to re-add the memory with the desired radix, or change the display radix from the Memory window Properties dialog.

- -radixenumnumeric

  (optional) Causes Verilog and SystemC enums to be displayed as numbers (formatted by the current radix). This overrides the default behavior of always showing enums symbolically.

- -radixenumsymbolic

  (optional) Restores the default behavior of displaying Verilog and SystemC enums as symbols by reversing the action of the -radixenumnumeric option.

- -wordsperline <num>

  (optional) Specifies how many words are displayed on each line in the memory window. By default, the information displayed will wrap depending on the width of the window.

  num — Any positive integer

- <object_name> ...

  (required) Specifies the hierarchical path of the memory to be displayed. Multiple memories are specified as a space separated list. Must be specified as the final argument to the add memory command.

  Wildcard characters are allowed.

Note

The *WildcardFilter* Tcl preference variable identifies types to ignore when matching objects with wildcard patterns.)

Related Topics

- Memory List Window
- Using the WildcardFilter Preference Variable

add message

This command is used within a DO file or script and specifies a user defined runtime message that is sent to the transcript and *.wlf* files. Messages are displayed in the Message Viewer window in the GUI. Refer to "Message Viewer Window" for information.

Syntax

add message **<message_body>** [-category <category>] [-efftime <time>] [-file <filename>] [-id <id_number>] [-inline] [-line <linenumber>] [-noident] [-nolevel] [-objects <list>] [-region region] [-severity {error | note | warning}]

Arguments

- <message_body>

  (required) User specified message.

- -category <category>

  (optional) Sets the category for the message in the Message Viewer window where the default is USER. The Message Viewer window Category column recognizes the following keywords:

Table 2-2. Message Viewer Categories

DISPLAY	FLI	PA
PLI	SDF	TCHK
VCD	VITAL	WLF
MISC	USER	<user-defined>

- -efftime <time>

  (optional) Specifies the simulation time when the message is saved to the log file. The time specified is listed in the Message Viewer window Time column when the message is called. Useful for placing messages at specific times in the simulation.

  <time> — Specified as an integer or decimal number.

- -file <filename>

  (optional) Displays a user specified string in the File Info column of the Message Viewer window.

- -id <id_number>

  (optional) Assigns an identification number to the message.

  <id_number> — Any non-negative integer from 0 - 9999 where the default is 0. The number specified is added to the base identification number of 80000.

- -inline

  (optional) Causes the message to also be returned to the caller as the return value of the add message command.

- -line <linenumber>

  (optional) Displays the user specified number in File Info column of the Message Viewer window.

- -noident

  (optional) Prevents return of the ID number of the message.

- -nolevel

  (optional) Prevents return of the severity level of the message.

- -objects <list>

  (optional) List of related design items shown in the Objects column of the Message Viewer window.

  <list> — A space separated list enclosed in curly braces ({}) or quotation marks (" ").

- -region region

  (optional) Message is displayed in the Region column of the Message Viewer window.

- -severity {error | note | warning}

  (optional) Sets the message severity level.

  error — Questa SIM cannot complete the operation.

  note — (default) The message is informational.

  warning — There may be a problem that will affect the accuracy of the results.

Examples

- Create a message numbered 80304.

add message -id 304 <message>

Related Topics

- displaymsgmode modelsim.ini variable
- msgmode modelsim.ini variable
- Message Viewer Window

add schematic

This command adds the specified process, signal, net, or register to the Incremental or Full view of the Schematic window. Wildcards are allowed.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

add schematic **<object> ...** [-connect <source_net> <destination_net>] [-incr | -full] {[-in] [-out] [-inout] | [-ports]} [-internal] [-nofilter] [-recursive] [-window <wname>]

Arguments

- <object> ...

  (required unless specifying -connect) Specifies a process, signal, net, or register to add to the Schematic window. Wildcards are allowed. Multiple objects are specified as a space separated list, Refer to the section "Wildcard Characters" for wildcard usage as it pertains to the add commands. Must be specified as the first argument to the add schematic command.

- -connect <source_net> <destination_net>

  (optional) Computes and displays in the Schematic window all paths between two nets.

  <source_net> — The net that originates the path search.

  <destination_net> — The net that terminates the path search.

- -incr | -full

  (optional) Adds the specified process, signal, net, or register to the Incremental (-incr) or Full (-full) view of the Schematic window. If neither is specified, the object is displayed in the current default mode of the Schematic window. If the Schematic window is not yet opened, the default is -incr.

- -in

  (optional) Specifies to add ports of mode IN.

- -inout

  (optional) Specifies to add ports of mode INOUT.

- -out

  (optional) Specifies to add ports of mode OUT.

- -ports

  (optional) Specifies to add all ports. This switch has the same effect as specifying -in, -out, and -inout together.

- -internal

  (optional) Specifies to add internal (non-port) objects.

- -nofilter

  (optional) Specifies that the *WildcardFilter* Tcl preference variable be ignored when finding signals or nets.

  The *WildcardFilter* Tcl preference variable identifies types to ignore when matching objects with wildcard patterns.

- -recursive

  (optional) Specifies that the scope of the search is to descend recursively into subregions. If omitted, the search is limited to the selected region.

  You can specify -r as an alias to this switch.

- -window <wname>

  (optional) Adds the object(s) to the specified Schematic window.

  <wname> — the name of the schematic window, as shown in the window's tab.

  This switch is useful when you have multiple schematic windows open.

  You can open a new schematic window by entering:

  view schematic -new

Examples

- Add all objects in the design to the schematic window.

  add schematic -r /*

- Add all objects in the region to the schematic window.

  add schematic *

- Open a new Schematic window with "SFLOW" as its title, then add signals to it.

  set SFLOW [view schematic -new -title SFLOW]

  add schematic -window $SFLOW /top/mysignals

  The custom window title "SFLOW" is saved as a TCL variable, then called using the '$' prefix.

Related Topics

- Schematic Window
- Using the WildcardFilter Preference Variable

add testbrowser

This command adds *.ucdb* file(s) to the test management browser.

Syntax

add testbrowser <ucdb_filename> [<ucdb_filename>...]

Arguments

- <ucdb_filename> [<ucdb_filename>...]

  (required: at least one *.ucdb*) Specifies the name of the *.ucdb* file(s) to be added. Multiple filenames are specified as a space separated list.

  Wildcard characters are allowed. (Note that the *WildcardFilter* Tcl preference variable identifies types to ignore when matching objects with wildcard patterns.)

Related Topics

- "Verification Management Browser Window"
- Using the WildcardFilter Preference Variable

add watch

This command adds signals and variables to the Watch window in the Main window.

SystemC objects and user-defined buses may also be added.

Refer to "Wildcard Characters" for wildcard usage as it pertains to the add commands.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

add watch **<object_name> ...** [-radix <type>] [-radixenumnumeric | -radixenumsymbolic]

Arguments

- <object_name> ...

  (required) Specifies the name of the object to be added. Multiple objects are entered as a space separated list. Must be specified as the first argument to the add watch command.

  Wildcard characters are allowed. (Note that the *WildcardFilter* Tcl preference variable identifies types to ignore when matching objects with wildcard patterns.)

  Variables must be preceded by the process name. For example,

  ```
  add watch myproc/int1
  ```

- -radix <type>

  (optional) Specifies a user-defined radix. If you do not specify this switch, the command uses the global default radix.

  <type> — binary, ascii, unsigned, decimal, octal, hex, sfixed, symbolic, time, ufixed, and default.

  Refer to the radix command for information about sfixed and ufixed radix types.

  You can change the default radix for the current simulation using the radix command. You can change the default radix permanently by editing the DefaultRadix variable in the *modelsim.ini* file.

- -radixenumnumeric

  (optional) Causes Verilog and SystemC enums to be displayed as numbers (formatted by the current radix). This overrides the default behavior of always showing enums symbolically.

- -radixenumsymbolic

  (optional) Restores the default behavior of displaying Verilog and SystemC enums as symbols by reversing the action of the -radixenumnumeric option.

Related Topics

- Watch window
- Using the WildcardFilter Preference Variable
- Wildcard Characters
- DefaultRadix variable

add wave

This command adds the following objects to the Wave window:

- VHDL signals and variables
- Verilog nets and registers
- SystemVerilog class objects
- SystemC primitive channels (signals)
- Dividers and user-defined buses.

If no port mode is specified, this command will display all objects in the selected region with names matching the object name specification.

Refer to "Wildcard Characters" for wildcard usage as it pertains to the add commands.

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

add wave [-allowconstants] [-clampanalog {0 | 1}] [-color <standard_color_name>]
[-depth <level>] [[-divider [<divider_name> ...] [-expand <signal_name>]
[-filter <f> | -nofilter <f>] [-format <type> | -<format>]
[-group <group_name> [<sig_name1> ...]] [-height <pixels>]
{[-in] [-inout] [-out] | [-ports]} [-internal] [-label <name>] [-max <real_num>]
[-min <real_num>] [-mvcall] [-mvcovm] [-mvcreccomplete] [-noupdate]
[-numdynitem <int>] [-optcells] [-position <location>] [-queueends]
[-<radix_type> | -radix <type>] [-radixenumnumeric | -radixenumsymbolic] [-recursive]
[-startdynitem <int>] [-time] [-window <wname>] [**<object_name> ...**]
[**{<object_name> {sig1 sig2 ...}}**]

Arguments

- -allowconstants

  (optional) *For use with wildcard searches.* Specifies that constants matching the wildcard search should be added to the Wave window.

  By default, constants are ignored because they do not change.

- -clampanalog {0 | 1}

  (optional) Clamps the display of an analog waveform to the values specified by -max and -min. Specifying a value of 1 prevents the waveform from extending above the value specified for -max or below the value specified for -min.

  0 — not clamped

  1 — (default) clamped

- -color <standard_color_name>

  (optional) Specifies the color used to display a waveform.

  <standard_color_name> — You can use either of the following:

  standard X Window color name — enclose 2-word names in quotes ("), for example:

  -color "light blue"

  rgb value — for example:

  -color #357f77

- -depth <level>

  (optional) Restricts a recursive search, as specified with -recursive to a specified level of hierarchy.

  <level> — Any integer greater than or equal to zero. For example, if you specify -depth 1, the command descends only one level in the hierarchy.

- -divider [<divider_name> ...]

  (optional) Adds a divider to the Wave window. If you do not specify this argument, the command inserts an unnamed divider.

  <divider_name> ... — Specifies the name of the divider, which appears in the pathnames column. Multiple objects entered as a space separated list.

  When you specify more than one <divider_name> the command creates a divider for each name.

  You can begin a name with a space, but you must enclose the name within quotes (") or braces ({ }) You cannot begin a name with a hyphen (-).

- -expand <signal_name>

  (optional) Instructs the command to expand a compound signal immediately, but only one level down.

  <signal_name> — Specifies the name of the signal. This string can include wildcards.

- -filter <f> | -nofilter <f>

  (optional) Allows a one-time modification of the WildcardFilter in the command invocation. The add list command can take as many [-filter <f>] and [-nofilter <f>] arguments as you would like to specify. Valid filters, <f>, are exactly the same set of words that can be applied to the WildcardFilter. The filter(s) used during a command starts with the WildcardFilter and then applies the user specified filters, if any. The -filter values are added to the filter, the -nofilter values are removed from the filter. They are applied in the order specified so conflicts are resolved with the last specified wins.

- -format <type> | -<format>

  (optional) Specifies the display format of the objects. Valid entries are:

-format <type>	-<format>	Display Format

-format literal	-literal	Literal waveforms are displayed as a box containing the object value.
-format logic	-logic	Logic signals may be U, X, 0, 1, Z, W, L, H, or '-'.
-format analog-step	-analog-step	Analog-step changes to the new time before plotting the new Y.
-format analog-interpolated	-analog-interpolated	Analog-interpolated draws a diagonal line.
-format analog-backstep	-analog-backstep	Analog-backstep plots the new Y before moving to the new time.
-format event	-event	Displays a mark at every transition.

The way each state is displayed is specified by the logic type display preference (refer to "modelsim.ini Variables").

The Y-axis range of analog signals is bounded by -max and -min switches.

- -group <group_name> [<sig_name1> ...]

  (optional) Creates a wave group with the specified group_name.

  <group_name> — Specifies the name of the group. You must enclose this argument in quotes (") or braces ({ }) if it contains any spaces.

  <sig_name> ... — Specifies the signals to add to the group. Multiple signals are entered as a space separated list. This command creates an empty group if you do not specify any signal names.

- -height <pixels>

  (optional) Specifies the height of the waveform in pixels.

  <pixels> — Any positive integer.

- -in

  (optional) *For use with wildcard searches*. Specifies that the scope of the search is to include ports of mode IN if they match the object_name specification.

- -out

  (optional) *For use with wildcard searches*. Specifies that the scope of the search is to include ports of mode OUT if they match the object_name specification.

- -inout

  (optional) *For use with wildcard searches*. Specifies that the scope of the search is to include ports of mode INOUT if they match the object_name specification.

- -ports

  (optional) *For use with wildcard searches*. Specifies that the scope of the listing is to include ports of modes IN, OUT, or INOUT.

- -internal

  (optional) *For use with wildcard searches*. Specifies that the scope of the search is to include internal objects (non-port objects) if they match the object_name specification.

- -label <name>

  (optional) Specifies an alternative name for the signal being added. For example,

  add wave -label c clock

  adds the *clock* signal, labeled as "c".

  This alternative name is not valid in a force or examine command; however, it can be used in a search command with the wave option.

- -max <real_num>

  (optional) Specifies the maximum Y-axis data value to be displayed for an analog waveform. Used in conjunction with the -min switch; the value you specify for -max must be greater than the value you specify for -min.

  <real_num> — Any integer that is greater than the value specified for -min.

- -min <real_num>

  (optional) Specifies the minimum Y-axis data value to be displayed for an analog waveform. Used in conjunction with the -max switch; the value you specify for -min must be less than the value you specify for -max.

  <real_num> — Any integer that is less than the value specified for -max.

  For example, if you know the Y-axis data for a waveform varies between 0.0 and 5.0, you could add the waveform with the following command:

  add wave -analog -min 0 -max 5 -height 100 my_signal

Note

Although -offset and -scale are still supported, the -max and -min arguments provide an easier way to define upper and lower limits of an analog waveform.

- -mvcall

  (optional) Specifies the inclusion of all Questa Verification IP protocol transactions when a wildcard is used (for example, add wave -r /*). By default, Questa Verification IP transactions are sometimes excluded from view with normal wildcard usage.

- -mvcovm

  (optional) Specifies the inclusion of all UVM or OVM sequence transactions when a wildcard is used (for example, add wave -r /*). By default, UVM/OVM sequence transactions are sometimes excluded from view with normal wildcard usage.

- -mvcreccomplete

  (optional) Enables the logging of all Questa Verification IP transaction instances which have been recognized as completed, whether or not they become used as legal protocol. By default, only transactions that become used as legal protocol are visible, which can prevent the transaction instances of interest from being logged soon enough to observe an issue.

- -noupdate

  (optional) Prevents the Wave window from updating when a series of add wave commands are executed in series.

- -numdynitem <int>

  (optional) Specifies the number of child elements of a queue or dynamic array to display in the Wave window. For example, if the value 3 is specified, then only three elements will be displayed in the Wave window.

  <int> — Any non-negative integer from 0 to the number of elements of the specified queue or dynamic array.

- -optcells

  (optional) Specifies that optimized cell ports are visible when using wildcards. By default optimized cell ports are not selected even if they match the specified wildcard pattern.

- -position <location>

  (optional) Specifies where the command adds the signals.

  <location> — Can be any of the following:

  top — Adds the signals to the beginning of the list of signals.

  bottom | end — Adds the signals to the end of the list of signals.

  before | above — Adds the signals to the location before the first selected signal in the wave window.

  after | below — Adds the signals to the location after the first selected signal in the wave window.

  <integer> — Adds the signals beginning at the specified point in the list of signals.

- -queueends

  (optional) Adds a SystemVerilog queue to the Wave window and displays the first and last elements of the queue.

  <queue> — The relative or full path to a queue.

- -<radix_type>

  (optional) Specifies the radix type for the objects that follow in the command. Valid entries (or any unique abbreviations) are: binary, ascii, unsigned, decimal, octal, hex, sfixed, symbolic, time, ufixed, and default.

If no radix is specified for an enumerated type, the default radix is used. You can change the default radix for the current simulation using the radix command. You can change the default radix permanently by editing the DefaultRadix variable in the *modelsim.ini* file.

If you specify a radix for an array of a VHDL enumerated type, Questa SIM converts each signal value to 1, 0, Z, or X.

- -radix <type>

  (optional) Specifies a user-defined radix. The -radix <type> switch can be used in place of the -<radix_type> switch. For example, -radix hexadecimal is the same as -hex.

  <type> — binary, ascii, unsigned, decimal, octal, hex, sfixed, symbolic, time, ufixed, and default.

  Refer to the radix command for information about sfixed and ufixed radix types.

  This option overrides the global setting of the default radix (the DefaultRadix variable in the *modelsim.ini* file) for the current simulation only.

- -radixenumnumeric

  (optional) Causes Verilog and SystemC enums to be displayed as numbers (formatted by the current radix). This overrides the default behavior of always showing enums symbolically.

- -radixenumsymbolic

  (optional) Restores the default behavior of displaying Verilog and SystemC enums as symbols by reversing the action of the -radixenumnumeric option.

- -recursive

  (optional) For use with wildcard searches. Specifies that the scope of the search is to descend recursively into subregions.

  If you do not specify this switch, the search is limited to the selected region. You can use the -depth argument to specify how far down the hierarchy to descend.

- -startdynitem <int>

  (optional) Specifies the index of a queue or dynamic array from where the Wave window starts displaying the data. For example, if a queue has 10 elements and -startdynitem 3 is specified, the display starts from q[3].

  <int> — Any non-negative integer where 0 is the default.

- -time

  (optional) Use time as the radix for Verilog objects that are register-based types (register vectors, time, int, and integer types).

- -window <wname>

  (optional) Adds objects to a specified window. Used to specify a particular wave window when multiple wave windows exist.

  <wname> — The name of the wave window (for instance, wave2). Must be an existing window. Does not create a new window.

Use the view command with the -new option to create a new window.

- <object_name> ...

  (required unless specifying {<object_name> {sig1 sig2 ...}) Specifies the names of objects to be included in the Wave window. Must be specified as the final argument to the add wave command. Wildcard characters are allowed. Multiple objects are entered as a space separated list. Note that the *WildcardFilter* Tcl preference variable identifies types to ignore when matching objects with wildcard patterns.

  Variables may be added if preceded by the process name. For example,

  add wave myproc/int1

- {<object_name> {sig1 sig2 ...}}

  (required unless specifying <object_name>) Creates a user-defined bus with the specified object name containing the specified signals (sig1 and so forth) concatenated within the user-defined bus. Must be specified as the final argument to the add wave command.

  sig — A space-separated list of signals, enclosed in braces ({ }), that are included in the user-defined bus. The signals may be either scalars or various sized arrays as long as they have the same element enumeration type.

Note

You can also select **Wave > Combine Signals** (when the Wave window is selected) to create a user-defined bus.

Examples

- Display an object named *out2*. The object is specified as being a logic object presented in gold.

  add wave -logic -color gold out2

- Display a user-defined, hex formatted bus named *address*.

  add wave -hex {address {a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0}}

- Add all wave objects in the region.

  add wave *

- Add all wave input ports in the region.

  add wave -in *

- Create a user-defined bus named "mybus" consisting of three signals. *Scalar1* and *scalar2* are of type std_logic and *vector1* is of type std_logic_vector (7 downto 1). The bus is displayed in hex.

  add wave -hex {mybus {scalar1 vector1 scalar2}}

  Slices and arrays may be added to the bus using either VHDL or Verilog syntax. For example:

```
add wave {vector3(1)}
add wave {vector3[1]}
add wave {vector3(4 downto 0)}
add wave {vector3[4:0]}
```

- Add the object *vec1* to the Wave window using symbolic values, adds *vec2* in hexadecimal, and adds *vec3* and *vec4* in decimal.

```
add wave vec1 -hex vec2 -dec vec3 vec4
```

- Add a divider with the name "-Example-". Note that for this to work, the first hyphen of the name must be preceded by a space.

```
add wave -divider " -Example- "
```

- Add an unnamed divider.

```
add wave -divider
add wave -divider ""
add wave -divider {}
```

- Open a new Wave window with "SV_Signals" as its title, then add signals to it.

```
set SV_Signals [view wave -new -title SV_Signals]
add wave -window $SV_Signals /top/mysignals
```

  The custom window title "SV_Signals" is saved as a TCL variable, then called using the ’$’ prefix.

Related Topics

- add list
- log
- Concatenation Directives
- Extended Identifiers
- Using the WildcardFilter Preference Variable
- Refer to Wave Window for more information on analog formats of waveform signals.

add_cmdhelp

This command adds the specified command name, description, and command arguments to the command-line help. You can then access the information using the help command.

To delete an entry, invoke the command with an empty command description and arguments. See examples.

The arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

add_cmdhelp **{<command_name>} {<command_description>} {<command_arguments>}**

Arguments

- {<command_name>}

  (required) Specifies the command name that will be entered as an argument to the help command. Must be enclosed in braces ({ }). The command_name must not interfere with an already existing command_name. Must be specified as the first argument to the add_cmdhelp command.

- {<command_description>}

  (required) Specifies a description of the command. Must be enclosed in braces ({ }). Must be specified as the second argument to the add_cmdhelp command.

- {<command_arguments>}

  (required) A space-separated list of arguments for the command. Must be enclosed in braces ({ }). If the command doesn't have any arguments, enter {}. Must be specified as the third argument to the add_cmdhelp command.

Examples

- Add a command named "date" with no arguments.

  add_cmdhelp date {Displays date and time.} {}

  Entering:

  VSIM> help date

  Returns:

  ```
  Displays date and time.
  Usage: date
  ```

- Add the change date command.

  add_cmdhelp {change date} {Modify date or time.} {-time|-date <arg>}

  Entering:

  VSIM> help change date

Returns:

```
Modify data or time.
Usage: change date -time|-date <arg>
```

- Deletes the change date command from the command-line help.

  add_cmdhelp {change date} {} {}

add_menu

This command adds a menu to the menu bar of the specified window, using the specified menu name. Use the add_menuitem, add_separator, add_menucb, and add_submenu commands to complete the menu.

Returns the full Tk pathname of the new menu.

Color and other Tk properties of the menu may be changed, after creating the menu, using the Tk menu widget configure command.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

add_menu **<window_name> <menu_name>** [<shortcut> [-hide_menubutton]]

Arguments

- <window_name>

  (required) Specifies the Tk path of the window to contain the menu. Must be specified as the first argument to the add_menu command.

  To add a menu to the Main window you must express this value as: "". For example,

  add_menu "" mymenu

  To add a menu to any other window, you need to determine the window_name by entering the view command. For example:

  view wave

  Returns:

  ```
  # .main_pane.wave
  ```

  Note that the <window_name> for each window, except the Main window, begins with a period (.).

- <menu_name>

  (required) Specifies the name to be given to the Tk menu widget. Must be specified as the first argument to the add_menu command.

- <shortcut>

  (optional) An integer that specifies the number of the letter in the menu name that is to be used as the shortcut where the default is -1, no shortcut. Numbering starts with 0 (first letter = 0, second letter = 1, third letter = 2, and so on). Optional unless you specify -hide_menubutton, in which case <shortcut> is required. Default is "-1", indicating no shortcut is to be used.

- -hide_menubutton

  (optional) Specifies that the new menu is not to be displayed. You can add the menu later by calling tk_popup on the menu path widget. Note that you must specify <shortcut> if you specify -hide_menubutton.

Examples

The following Tcl code is an example of creating user-customized menus. It adds a menu containing a top-level item labeled "Do My Own Thing...", which prints "my_own_thing.signals", and adds a cascading submenu labeled "changeCase" with two entries, "To Upper" and "To Lower", which echo "my_to_upper" and "my_to_lower" respectively. A checkbox that controls the value of myglobalvar (.signals:one) is also added.

```
set myglobalvar 0
set wname [view wave]; #Gets path to Wave window
proc AddMyMenus {wname} {
    global myglobalvar
    set cmd1 "echo my_own_thing $wname"
    set cmd2 "echo my_to_upper $wname"
    set cmd3 "echo my_to_lower $wname"

#               WindowName  Menu               MenuItem label        Command
#               ----------  ----               --------------------  -------
add_menu        $wname      mine
add_menuitem    $wname      mine               "Do My Own Thing..."  $cmd1
add_separator   $wname      mine              ;#------------------   -------
add_submenu     $wname      mine               changeCase
add_menuitem    $wname      mine.changeCase    "To Upper"            $cmd2
add_menuitem    $wname      mine.changeCase    "To Lower"            $cmd3
add_submenu     $wname      mine               vars
add_menucb      $wname      mine.vars          "Feature One"         -variable
                                                                        myglobalvar
                                                                     -onvalue 1
                                                                     -offvalue 0
                                                                     -indicatoron 1
}
AddMyMenus $wname
```

This example is available in the following DO file:

```
<install_dir>/examples/gui/addmenu/addmenu.do
```

You can run the DO file to add the "Mine" menu shown in the illustration, or modify the file for different results.

To execute the DO file, select **Tools > Execute Macro** (Main window), or use the **do** command.

Related Topics

- add_menucb
- add_separator
- change_menu_cmd
- add_menuitem
- add_submenu
- User Defined Buttons and Menus

add_menucb

This command creates a checkbox within the specified menu of the specified window. A checkbox is a small box with a label. Clicking on the box will toggle the state, from on to off or the reverse.

When the box is "on", the Tcl global variable <var> is set to <onval>. When the box is "off", the global variable is set to <offval>. Also, if something else changes the global variable, its current state is reflected in the state of the checkbox. Returns nothing.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

add_menucb **<window_name> <menu_name> <text>** [**-indicatoron {0 | 1}**]
[**-onvalue <onval>**] [**-offvalue <offval>**] [**-variable <var>**]

Arguments

- <window_name>

  (required) Specifies the Tk path of the window to contain the menu. Must be specified as the first argument to the add_menucb command.

  To add a menu to the Main window you must express this value as: "". For example:

  add_menucb "" mymenu

  To add a menu to any other window, you must determine the window_name by executing the **view** command, for example:

  view wave

  Returns:

  ```
  # .main_pane.wave
  ```

  Note that all windows, other than the Main window, begin with a period (.).

- <menu_name>

  (required) Specifies the name of the Tk menu widget. Must be specified as the second argument to the add_menucb command.

- <text>

  (required) Specifies the text to be displayed next to the checkbox. Must be specified as the third argument to the add_menucb command.

- -indicatoron {0 | 1}

  (optional) Specifies whether or not the status indicator is displayed.

  0 — off

  1 — (default) on

- -onvalue <onval>

  (optional) Specifies the value of the global Tcl variable when the box is "on".

  <onval> — A value specific to the global Tcl variable specified in -variable.
- -offvalue <offval>

  (optional) Specifies the value of the global Tcl variable when the box is "off".

  <offval> — A value specific to the global Tcl variable specified in -variable.
- -variable <var>

  (optional) Specifies the global Tcl variable to be reflected and changed.

  <var> — A global Tcl variable.

Examples

```
add_menucb $wname mine.vars "Feature One" \
-variable myglobalvar($wname:one) -onvalue 1 -offvalue 0 -indicatoron 1
```

This command is also used as part of the **add_menu** example.

Related Topics

- add_menu
- add_menuitem
- add_separator
- add_submenu
- change_menu_cmd

add_menuitem

This command creates a menu item within the specified menu of the specified window. May be used within a submenu.

Returns nothing.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

add_menuitem **<window_name> <menu_path> <text> <cmd>** [<shortcut>]

Arguments

The arguments <window_name>, <menu_path>, <text>, and <cmd> are order dependent and must be entered as specified in the Syntax section.

- <window_name>

  (required) Specifies the Tk path of the window to contain the menu. Must be specified as the first argument to the add_menuitem command.

  To add a menu to the Main window you must express this value as: "". For example,

  ```
  add_menu "" mymenu
  ```

  To add a menu to any other window, you must determine the window_name by executing the view command, for example:

  ```
  view wave
  ```

  Reurns:

  ```
  # .main_pane.wave
  ```

  Note that all windows, other than the Main window, begin with a period (.).

- <menu_path>

  (required) Specifies the name of the Tk menu widget plus submenu path. Must be specified as the second argument to the add_menuitem command.

- <text>

  (required) Specifies the text to be displayed. Must be specified as the third argument to the add_menuitem command.

- <cmd>

  (required) Specifies the command to be executed when the menu item is selected with the left mouse button. Must be specified as the fourth argument to the add_menuitem command.

  To echo the command and display the return value in the Main window, prefix the command with the transcribe command. The transcribe command will also echo the results to the Transcript window.

- <shortcut>

  (optional) An integer that specifies the number of the letter in the menu name that is to be used as the shortcut where the default is -1, no shortcut. Numbering starts with 0 (first letter = 0, second letter = 1, third letter = 2, and so on).

Examples

```
add_menuitem $wname user  "Save Results As..."  $my_save_cmd
```

This command is also used as part of the add_menu example.

Related Topics

- add_menu
- add_menucb
- add_separator
- add_submenu
- change_menu_cmd

add_separator

This command adds a separator as the next item in the specified menu path in the specified window.

Returns nothing.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

add_separator **<window_name> <menu_path>**

Arguments

The arguments <window_name> and <menu_path> are order dependent and must be entered as specified in the Syntax section.

- <window_name>

  (required) Specifies the Tk path of the window to contain the menu. Must be specified as the first argument to the add_separator command.

  To add a menu to the Main window you must express this value as: "". For example,

  add_menu "" mymenu

  To add a menu to any other window, you must determine the window_name by executing the **view** command, for example:

  view wave

  Returns:

  ```
  # .main_pane.wave
  ```

  Note that all windows, other than the Main window, begin with a period (.).

- <menu_path>

  (required) Specifies the name of the Tk menu widget plus submenu path. Must be specified as the second argument to the add_separator command.

Examples

add_separator $wname user

This command is also used as part of the **add_menu** example.

Related Topics

- add_menu
- add_menucb
- add_menuitem
- add_submenu
- change_menu_cmd

add_submenu

This command creates a cascading submenu within the specified menu path of the specified window. May be used within a submenu.

Returns the full Tk path to the new submenu widget.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

add_submenu **<window_name> <menu_path> <name>** [<shortcut>]

Arguments

The arguments <window_name>, <menu_path>, and <name> are order dependent and must be specified as shown in the Syntax section.

- <window_name>

  (required) Specifies the Tk path of the window to contain the menu. Must be specified as the first argument to the add_submenu command.

  To add a menu to the Main window you must express this value as: "". For example,

  add_menu "" mymenu

  To add a menu to any other window, you must determine the window_name by executing the view command, for example:

  view wave

  Returns:

  ```
  # .main_pane.wave
  ```

  Note that all windows, other than the Main window, begin with a period (.).

- <menu_path>

  (required) Specifies the name of the Tk menu widget plus submenu path. Must be specified as the second argument to the add_submenu command.

- <name>

  (required) Specifies the name to be displayed on the submenu. Must be specified as the third argument to the add_submenu command.

- <shortcut>

  (optional) An integer that specifies the number of the letter in the menu name that is to be used as the shortcut where the default is -1, no shortcut. Numbering starts with 0 (first letter = 0, second letter = 1, third letter = 2, and so on).

Examples

This command is used as part of the add_menu example.

Related Topics

- add_menu
- add_menucb
- add_menuitem
- add_separator
- change_menu_cmd

alias

This command displays or creates user-defined aliases. Any arguments passed on invocation of the alias will be passed through to the specified commands.

Returns nothing. Existing commands (for example, run, env, and so forth) cannot be aliased.

Syntax

alias [<name> ["<cmds>"]]

Arguments

- <name>

  (optional) Specifies the new procedure name to be used when invoking the commands.

- "<cmds>"

  (optional) Specifies the command or commands to be evaluated when the alias is invoked. Multiple commands are specified as a semicolon (;) separated list. You must enclose the string in quotes (“”).

Examples

- List all aliases currently defined.

  alias

- List the alias definition for the specified name if one exists.

  alias <name>

- Create a Tcl procedure, "myquit", that when executed, writes the contents of the List window to the file *mylist.save* by invoking **write list**, and quits Questa SIM by invoking **quit**.

  alias myquit "write list ./mylist.save; quit -f"

archive load

The archive load command allows you to load an archived debug database (.dbar) file that was previously created with the archive write command. The archived file may include a number of WLF files, design source files, and a DBG file.

Syntax

archive load <archive_name> [-dbgDir <directory_name>] -wlf <wlf_file_name>

Arguments

- <archive_name>

  (required) Specifies the name of the archived file to be opened for reading. A suggested suffix is .dbar.

- -dbgDir <directory_name>

  (optional) Specifies a location to extract files into. Files are extracted on-demand when Questa SIM needs them. If you do not specify this switch, the command extracts to the current working directory.

- -wlf <wlf_file_name>

  (required) Specifies the WLF files to open for analysis.

  <wlf_file_name> — can be a single file or a list of files. A list of file names must be enclosed in curly braces {}. The name of the wlf file must be exactly the same as that specified in the archive write command, including the pathname, if provided.

Related Topics

- archive write

archive write

The archive write command allows you to create a debug archive file, with the file extension .dbar, that contains one or more WLF files, debug information captured from the design library, an optional connectivity debug database file, and optional HDL source files. With this archived file, you can perform post-simulation debugging in different location from that which the original simulation was run.

Syntax

archive write <archive_name> -wlf <wlf_file_name> [-include_src] [-dbg <dbg_file_name>]

Arguments

- <archive_name>

  (required) Specifies the name of the archive file to be created. A suggested suffix is .dbar.

- -wlf <wlf_file_name>

  (required) Specifies the name of the WLF file to use for post-simulation analysis.

  <wlf_file_name> — can be a single file or a list of files enclosed in curly braces {} if you want to capture more than one WLF file in the archive.

- -include_src

  (optional) Indicate if source files should be captured in the archive. This is off by default, which means no source will be in the archive.

- -dbg <dbg_file_name>

  (optional) Specifies the name of an existing debug database (.dbg) file to be included in the archive.

assertion action

This command allows you to set the assertion action for concurrent assertion starts, failures, passes, and antecedent matches. The actions can be: continue, break, exit, or a tcl subroutine call. The action setting for start, antecedent and pass conditions works only if:

- the assertion is browseable – for example, the +acc=a argument is used with the vopt command.
- Assertion debug is set – for example, the -assertdebug switch is used with the vsim command).

You can define a tcl subroutine in a *.do* file and call it on a specific assertion event.

Syntax

assertion action
-cond [start | antecedent | pass | fail | threads <num_of_threads> | global_fail_limit <value>]
-exec [continue | break | exit | tcl_subroutine | kill | off]
-actionblock [passon | passoff | failon | failoff | nonvacuouson | vacuousoff]
[-assert | -cover] [-filter <pattern>] [-recursive]
[-severity info | note | warning | error | failure | fatal]
<path> [<path> ...]

Arguments

- -cond [start | antecedent | pass | fail | threads <num_of_threads> | global_fail_limit <value>]

  (required) Designates the assertion condition as start, antecedent, pass, fail, threads, or global fail limit.

  start — When a new assertion attempt starts.

  pass — When an assertion passes.

  fail — When an assertion fails.

  antecedent — When the antecedent (left hand side) of an implication succeeds.

  threads <num_of_threads> — Used in conjunction with -exec [kill | off] to designate the limit of the number of assertion or cover directive threads. Default number of threads is unlimited. This limit can be changed using the AssertionThreadLimit or CoverThreadLimit variables, set in the *modelsim.ini* file.

  global_fail_limit <value> — When a total fail count (<value>) for all assertions in the design is reached. Can be used with -severity to set actions for global fail limits based on severity of assertions.

- -exec [continue | break | exit | tcl_subroutine | kill | off]

  (required) Specifies the type of action to take on the designated assertion condition.

  continue — (default) No action taken. This is the default value if you do not specify this switch.

  break — Halt simulation and return to the Questa SIM prompt.

exit — Halt simulation and exit Questa SIM.

tcl_subroutine — Execute tcl subroutine.

kill — Used only with -cond threads <num_of_threads> to designate the action to be taken when the thread count limit is reached. When limit is reached, all existing threads are terminated and no new attempts are initiated.

off — Used only with -cond threads <num_of_threads> to designate the action to be taken when the threadcount limit is reached. When limit is reached, all existing thread attempts continue evaluating, however no new attempts are initiated.

- -actionblock [passon | passoff | failon | failoff | nonvacuouson | vacuousoff]

  (optional) Controls the execution of assertions in action blocks. The '-cond' and '-exec' arguments can not be combined with this argument. The following options are provided to control the execution of action blocks:

  passon - Enables execution of pass action for vacuous and nonvacuous passes.

  passoff - Disables execution of pass action for vacuous and nonvacuous passes.

  failon - Enables execution of fail action for failures.

  failoff - Disables execution of fail action for failures.

  nonvacuouson - Enables execution of pass action for nonvacuous passes.

  vacuousoff - Disables execution of pass action for vacuous passes.

- -assert | -cover

  (optional) The -assert switch specifies that the action applies only to assertions. The -cover switch specifies that the action applies only to cover directives. If neither is specified, the action applies to both assertions and cover directives.

- -filter <pattern>

  (optional) Specifies that the assertion action arguments affect only the assertions whose names match the specified pattern.

  <pattern> — Specifies the character(s) to be matched in the search. Wildcards are permitted.

- -recursive

  (optional) For use with wildcard matching. Specifies that the scope of the matching is to descend recursively into subregions. If omitted, the search is limited to the selected region. Applies to all paths specified in the command. You can abbreviate this switch to “-r”.

- -severity info | note | warning | error | failure | fatal

  (optional) Specifies the assertion severity level. When -severity is specified only assertions with the same or higher severity are selected. If not specified, assertions of all severities are selected. Can be used to set actions for global fail limits based on severity of assertions.

- <**path**> [<path> ...]

  (required) A space separated list of paths, that specifies the assertions to be affected. Multiple paths and wildcards are allowed. The path specifies assertions or a design region containing multiple assertions. Must be specified as the final argument to the **assertion active** command.

  You must specify at least one <path> argument, but can also specify more in a space separated list.

Examples

- If the following command is used:

  assertion action -cond global_fail_limit 100 -exec break

  a simulation break will occur when the total number of failed assertions reaches 100.

Related Topics

- assertion active
- assertion count
- assertion enable
- assertion fail
- assertion pass
- assertion profile
- atv log
- Viewing Assertions in the Assertions Window

assertion active

This command instructs the simulator to report on any active assertion directives at the end of simulation (EOS). Active assertion directives will be indicated in the Assertions tab of the Analysis window with the text "active at end of simulation" in the EOS Note column. If the PSL assert directive is strong, the EOS Note column will report both the usual "active at end of simulation" note along with a PSL strong error message.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

assertion active [-eosnote {off | on}] [-filter <pattern>] [-recursive]
[-severity info | note | warning | error | failure | fatal] **<path> [<path> ...]**

Arguments

- -eosnote {off | on}

  (optional) Controls the reporting of active assertion directives in the EOS Note column of the Assertions tab of the Analysis window. The EOS Note message scopes will match those that appear in assertion messages during runtime.

  off — (default) Reporting of active assertion directives turned off.

  on — Reporting of active assertion directives turned on.

- -filter <pattern>

  (optional) Limits EOS reporting to only those assertion directives that are active and whose names match the specified pattern.

  <pattern> — Specifies the character(s) to be matched in the search. Wildcards are permitted.

- -recursive

  (optional) *For use with wildcard matching*. Specifies that the scope of the matching is to descend recursively into subregions. If omitted, the search is limited to the selected region. Applies to all paths specified in the command. You can abbreviate this switch to “-r”.

- -severity info | note | warning | error | failure | fatal

  (optional) Specifies the assertion severity level. When -severity is specified only assertions with the same or higher severity are selected. If not specified, assertions of all severities are selected.

- <path> [<path> ...]

  (required) A space separated list of paths, that specifies the assertions to be affected. Multiple paths and wildcards are allowed. The path specifies assertions or a design region containing multiple assertions. Must be specified as the final argument to the assertion active command.

You must specify at least one <path> argument, but can also specify more in a space separated list.

Examples

assertion active -eosnote on -r /*

assertion active -eosnote off /tb/assert*

assertion active -eosnote on /tb/assert* /tb/cntrl/*

Related Topics

- assertion action
- assertion count
- assertion enable
- assertion fail
- assertion pass
- assertion profile
- atv log
- Viewing Assertions in the Assertions Window

assertion count

This command returns the sum of the assertion failure counts for the specified set of assertion directive instances. Returns a "No matches" warning if the given path does not contain any assertions.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

assertion count {**-fails | -failattempts**} [-lang sva | psl | vhdl] [-concurrent | -immediate] [-filter <pattern>] [-severity info | note | warning | error | failure | fatal] [-recursive] **<path> [<path> ...]**

Arguments

- -fails | -failattempts

  (required) Controls reporting of assertion directive failures. Must be specified as the first argument to the assertion count command.

  -fails — Returns the sum of the given assertion directive instances that have a non-zero fail count.

  -failattempts — Returns the sum of all of the fail counts of the specified assertion directive instances.

- -lang sva | psl | vhdl

  (optional) Specifies assertions of a specific language (SVA, PSL, or VHDL). You may specify multiple languages by using this option multiple times. If -lang is not specified, all three languages are selected.

- -concurrent | -immediate

  (optional) Only concurrent assertions are selected when the -concurrent switch is used. Only immediate assertions are selected when the -immediate switch is used. These are mutually exclusive options. If neither switch is specified, both concurrent and immediate assertions are selected.

- -filter <pattern>

  (optional) Specifies that the command affects only the assertions whose names match the specified pattern.

  <pattern> — Specifies the character(s) to be matched in the search. Wildcards are permitted.

- -severity info | note | warning | error | failure | fatal

  (optional) Specifies the assertion severity level. When -severity is specified only assertions with the same or higher severity are selected. If not specified, assertions of all severities are selected.

- -recursive

  (optional) *For use with wildcard matching*. Specifies that the scope of the matching is to descend recursively into subregions. If omitted, the search is limited to the selected region. Applies to all paths specified in the command. You can abbreviate this switch to “-r”.

- <path> [<path> ...]

  (required) A space separated list of paths, that specifies the assertions to be affected. Multiple paths and wildcards are allowed. The path specifies assertions or a design region containing multiple assertions. Must be specified as the final argument to the assertion count command.

Examples

```
assertion count -fails -r /
```

Related Topics

- assertion action
- assertion active
- assertion enable
- assertion fail
- assertion pass
- assertion profile
- atv log
- Configuring Assertions

assertion enable

This command enables and disables assertions and cover directives.

The assertion enable command introduces two levels of command function. If you use the -force option, this command takes precedence over any use of assertion enable where -force is not used, or over any use of the $asserton/$assertoff/$assertkill system tasks. With -force, the command is a first-level command.

Without -force, the command is a second-level command - that is, it is the same level as the $asserton/$assertoff/$assertkill system tasks. Second-level commands override each other depending on which is issued last.

The -release option returns the status of the assertions specified by the path to the status based on last issued second level command, whether it was issued before or after -force. (See Example.)

Arguments to this command are order-dependent. The <path> must be specified as the final argument. Please read through the argument descriptions for more information.

Syntax

assertion enable **-on | -off | -release** [-assert | -cover] [-concurrent | -immediate] [-filter <pattern>] [-force] [-lang {sva | vhdl | psl}]... [-recursive] [-severity info | note | warning | error | failure | fatal] **<path> [<path> ...]**

Arguments

- -on | -off | -release

  (required) Controls assertions and cover directives. Must be specified as the first argument to the assertion enable command.

  -on — Enables assertions and cover directives specified by <path>.

  -off — Disables

  -release — Returns the status of the assertions (specified by <path>) to the status based on last issued second level command, whether it was issued before or after the use of -force with assertion enable. (Refer to the Examples section.)

- -assert | -cover

  (optional) Specifies that the command will only apply to assertions (-assert) or to cover directives (-cover). If not specified, the command applies to both.

- -concurrent | -immediate

  (optional) Specifies that the command will only apply to concurrent or to immediate assertions. If not specified, the command will apply to both.

- -filter <pattern>

  (optional) Enables only those assertions whose names match the specified pattern.

<pattern> — Specifies the character(s) to be matched in the search. Wildcards are permitted.

- -force

  (optional) Forces the command to take precedence over any other invocation of the command, or of any use of the $asserton/$assertoff/$assertkill system task. (Refer to Example.)

- -lang {sva | vhdl | psl}

  (optional) Specifies that the command applies only to assertions and cover directives of the designated language. Can be specified multiple times to choose more than one language.

- -recursive

  (optional) *For use with wildcard matching*. Specifies that the scope of the matching is to descend recursively into subregions. If omitted, the search is limited to the selected region. Applies to all paths specified in the command. You can abbreviate this switch to “-r”.

- -severity info | note | warning | error | failure | fatal

  (optional) Specifies the assertion severity level. When -severity is specified only assertions with the same or higher severity are selected. If not specified, assertions of all severities are selected.

- <path> [<path> ...]

  (required) A space separated list of paths, that specifies the assertions to be affected. Multiple paths and wildcards are allowed. The path specifies assertions or a design region containing multiple assertions. Must be specified as the final argument to the assertion enable command.

Examples

- Understanding first-level and second-level commands and the use of -force and -release.

  If the following command is used:

  assertion enable -on -r assert1 /tb/assert* /tb/cntrl/*

  an invocation of the $asserton/$assertoff/$assertkill system task can disable the assertion named *assert1* in the following locations: /tb/assert* and /tb/cntrl/.

  If the following command is used:

  assertion enable -on -force -r assert1 /

  an invocation of the $asserton/$assertoff/$assertkill system task cannot disable the *assert1* assertion, from the path /.

  If the following command is used:

  assertion enable -off -r assert1

  it has no affect on the *assert1* assertion because -force is in effect.

If the following command is used:

assertion enable -release -r assert1

the -force option is released and the *assert1* assertion is disabled due to the use of a previous -off option.

If the following command is used:

assertion enable -off -force -r assert1

the *assert1* assertion is disabled with the -force option.

If the following system task is used:

$asserton();

it has no affect right away due to the use of the -force option.

If the following command is used:

assertion enable -release -r assert1

the *assert1* assertion is enabled due to the use of $asserton().

If the following command is used:

assertion enable -off -force assert1

the *assert1* assertion is disabled

If the following command is used:

assertion enable -release assert1

the assert1 is enabled due to previous $asserton. The effect of -force is released.

Related Topics

- assertion action
- assertion active
- assertion count
- assertion fail
- assertion pass
- assertion profile
- atv log
- Configuring Assertions

assertion fail

This command configures simulator behavior in response to a SystemVerilog or PSL assertion failure.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

assertion fail [-action {continue | break | exit}] [-filter <pattern>] [-limit {none | <count>}] [-log {on | off}] [-lvlog] [-recursive] **<path> [<path> ...]**

Arguments

- -action {continue | break | exit}

  (optional) Specifies the action to take when an assertion fails. You can specify this option multiple times, where it applies to all paths that follow it in the command line.

  continue — (default) No action taken. This is not the same as disabling an assertion since logging may still be enabled for the directive. This is the default value if you do not specify this switch.

  break — Halt simulation and return to the Questa SIM prompt.

  exit — Halt simulation and exit Questa SIM.

  You can change the permanent default by setting the AssertionFailAction variable in the *modelsim.ini* file.

- -filter <pattern>

  (optional) Specifies that the assertion fail arguments affect only the assertions whose names match the specified pattern.

  <pattern> — Specifies the character(s) to be matched in the search. Wildcards are permitted.

- -limit {none | <count>}

  (optional) Sets a limit on the number of times Questa SIM responds to an assertion failing.

  none — No limit; failure tracking remains enabled for the duration of the simulation. This is the default behavior if you do not specify this switch.

  <count> — Specify a whole number.

  Once the limit is reached for a particular assertion, Questa SIM disables that assertion. Questa SIM continues to respond to others if their limit has not been reached. You can change the permanent default by setting the AssertionLimit variable in the *modelsim.ini* file.

- -log {on | off}

  (optional) Specifies whether to write a transcript message when an assertion fails. This option may be specified multiple times; it applies to all paths that follow it in the command line.

on — (default) Enable transcript logging for all types of assertion failures (SVA and VHDL).

off — Disable transcript logging.

You can change the permanent default by setting the AssertionFailLog variable in the *modelsim.ini* file.

SystemVerilog assertion messages coming from action blocks are controlled by severity system tasks ($fatal, $error, $warning and $info). You can choose to ignore these system tasks using the GUI or by setting the following variables in the *modelsim.ini* file: IgnoreSVAError, IgnoreSVAFatal, IgnoreSVAInfo, and IgnoreSVAWarning.

- -lvlog

  (optional) Local variable values corresponding to failed assertion threads are printed to the Transcript when you run vsim -assertdebug. You can change the permanent default (on) by setting the AssertionFailLocalVarLog variable in the *modelsim.ini* file.

- -recursive

  (optional) *For use with wildcard matching*. Specifies that the scope of the matching is to descend recursively into subregions. If omitted, the search is limited to the selected region. Applies to all paths specified in the command.

- <**path**> [<path> ...]

  (required) A space separated list of paths, that specifies the assertions to be affected. Multiple paths and wildcards are allowed. The path specifies assertions or a design region containing multiple assertions. Must be specified as the final argument to the assertion fail command.

Examples

- Disable logging for assertions *a.b.c.assert__0* and *a.b.c.assert__1*. The -log argument applies to all paths that follow it on the command line.

  assertion fail -log off a.b.c.assert__0 a.b.c.assert__1

- Disable logging for assertion *a.b.c.assert__0* but enables it for *a.b.c.assert__1*.

  assertion fail -log off a.b.c.assert__0 -log on a.b.c.assert__1

- Set the failure response limit to 4 for all assertions in *mydesign*. Each assertion failure will be responded to a maximum of 4 times during the current simulation.

  assertion fail -r / -limit 4 mydesign

Related Topics

- assertion action
- assertion active
- assertion count
- assertion enable
- assertion pass
- assertion profile
- atv log
- Configuring Assertions

assertion pass

This command configures simulator behavior in response to a SystemVerilog or PSL assertion pass.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

assertion pass [-log {on | off}] [-filter <pattern>] [-recursive] **<path> [<path> ...]**

Arguments

- -log {on | off}

  (optional) Specifies whether to write a transcript message when a SystemVerilog or PSL assertion passes. This option may be specified multiple times; it applies to all paths that follow it in the command line.

  on — Enable transcript logging. (default)

  off — Disable transcript logging.

  You can change the permanent default by setting the AssertionPassLog variable in the *modelsim.ini* file.

  The -log argument does not apply to SystemVerilog assertion messages coming from action blocks. Those messages are always logged and are controlled by severity system tasks ($fatal, $error, $warning and $info). You can choose to ignore these system tasks using the GUI or by setting the following variables in the *modelsim.ini* file: IgnoreSVAError, IgnoreSVAFatal, IgnoreSVAInfo, and IgnoreSVAWarning.

- -filter <pattern>

  (optional) Specifies that the assertion pass arguments affect only the assertions whose names match the specified pattern.

  <pattern> — Specifies the character(s) to be matched in the search. Wildcards are permitted.

- -recursive

  (optional) *For use with wildcard matching*. A switch that specifies that the scope of the matching is to descend recursively into subregions. If omitted, the search is limited to the selected region. Applies to all paths specified in the command.

- <path> [<path> ...]

  (required) Specifies the assertions to be affected. Multiple paths are specified as a space separated list. Multiple paths and wildcards are allowed. The path specifies assertions or a design region containing multiple assertions. Must be specified as the final argument to the assertion pass command.

Examples

- Turn on logging for assertion *a.b.c.assert__0* but not *a.b.c.assert__1*.

```
assertion pass -log on a.b.c.assert__0 -log off a.b.c.assert_1
```

Related Topics

- assertion action
- assertion active
- assertion count
- assertion enable
- assertion fail
- assertion profile
- atv log
- Configuring Assertions

assertion profile

This command generates a fine grained profile of memory usage for assertions and cover directives including: current memory used, peak memory used (as well as the simulation run time at which it peaked), and the cumulative thread count for the assertion or cover directive.

This fine-grained memory usage profile data is reported in four columns of the Assertions Browser and Cover Directives Browser: current memory, peak memory, peak memory time, and cumulative threads.

If this command is used in the middle of the simulation, memory usage profiling starts|stops from that simulation time onwards. This command does not require that designs be compiled with +acc, or that vsim should be running in the -assertdebug mode. It can be given in fully optimized simulation run.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

assertion profile [-threadthreshold <number_of_threads>] **{on | off}**

Arguments

- -threadthreshold <number_of_threads>

  (optional) Causes the assertion engine to generate messages in the transcript anytime the number of threads in an assertion or cover directive exceeds the thread threshold at a clock edge. Only functions when assertion profiling is “on.”

  <number_of_threads> — Any positive integer.

- {on | off}

  (required) Enables or disables the collection of assertion and cover directive memory profile data. assertion profile on may be given at any time during simulation. Must be specified as the final argument to the assertion profile command.

  on — Collection of memory profile data enabled.

  off — Collection of memory profile data disabled.

Examples

- Thread threshold is reached.

  assertion profile -threadthreshold 100 on

  The transcript message produced when the threshold is reached is as follows:

```
# ** Note: Assertion thread threshold reached. Thread count = 110,
Memory = 4.8KB
#   Time: 215 ns  Scope: test.assert01 File: ./src/profile01.sv
Line: 9
# ** Note: Assertion thread threshold reached. Thread count = 110,
Memory = 4.8KB
#   Time: 215 ns  Scope: test.cover01 File: ./src/profile01.sv Line:
10
# ** Note: Assertion thread threshold reached. Thread count = 110,
Memory = 4.8KB
#   Time: 215 ns  Scope: test.seq File: ./src/profile01.sv Line: 3
```

Note

The message is generated at every clock edge for every assertion, cover directive, and endpoint whose current thread count is more than the threshold. It also shows the memory usage, which should be the same as that shown in the current memory column of the Assertions Browser and Cover Directives Browser.

Related Topics

- assertion action
- assertion active
- assertion count
- assertion enable
- assertion fail
- assertion pass
- assertion report
- atv log
- Configuring Assertions
- Assertions Window
- Cover Directives Window

assertion report

This command generates a status report for each SystemVerilog or PSL assertion matching the path specification.

Syntax

assertion report [-append] [-failed] [-file <filename>] [-filter <pattern>] [-lang psl|sva|vhdl] [-noconcurr] [-noimmed] [-profile] [-recursive] [-severity info|note|warning|error|failure|fatal] [-totalmemory] [-verbose] [-xml] <path>

Arguments

- -append

  (optional) Appends the current assertion statistics to the named output file (-file <filename>).

- -failed

  (optional) Displays only failed assertions in report.

- -file <filename>

  (optional) Specifies a file name for the report. Default is to write the report to the Transcript window. Environment variables may be used in the pathname.

- -filter <pattern>

  (optional) Specifies that the command affects only those objects whose name matches the specified pattern.

  <pattern> — Specifies the character(s) to be matched in the search. Wildcards are permitted.

- -lang psl|sva|vhdl

  (optional) Specifies assertions of a specific language (SV, PSL, or VHDL). You may specify multiple languages by using this option multiple times. If -lang is not specified, all three languages are selected.

- -noconcurr

  (optional) Disables reporting of any concurrent assertions.

- -noimmed

  (optional) Disables reporting of any immediate assertions.

- -profile

  (optional) Enables assertion profiling, which is displayed in the Assertions window.

- -recursive

  (optional) Recursively selects objects under the given scope.

- -severity info|note|warning|error|failure|fatal

  (optional) Specifies the assertion severity level. When -severity is specified only assertions with the same or lower severity are selected. If not specified, assertions of all severities are selected.

- -totalmemory

  (optional) Reports the total memory used by the assertion engine. This data is printed if assertions are being profiled. The profiling for assertion can be switched on/off using command 'assertion profile on/off'.

- -verbose

  (optional) Generates a verbose report showing full assertion information including other counts like the vacuous, attempted, disabled, and active counts.

- -xml

  (optional) Generates assertion report in xml format

- <path>

  (required) Specifies the path of an assertion object. Wildcards are permitted.

Related Topics

- assertion action
- assertion active
- assertion count
- assertion enable
- assertion fail
- assertion profile
- atv log
- Configuring Assertions
- Assertions Window
- Cover Directives Window

atv log

Prerequisites:

- Use the +acc=a argument with the vopt command or -voptargs=+acc=a argument with the **vsim** command
- Use the -assertdebug argument with the vsim command

This command enables or disables assertion thread viewing (ATV) for the specified assertion. Multiple assertions may be specified by their pathnames.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

atv log {**-enable | -disable**} [-asserts] [-covers] [-recursive] **<path> [<path> ...]**

Arguments

- -enable | -disable

  (required) Controls assertion thread viewing for the specified assertions. Must be specified as the first argument to the atv log command.

  -enable — Turns on assertion thread viewing.

  -disable — Turns off assertion thread viewing.

- -asserts

  (optional) Enables thread viewing for assertions only. Thread viewing for cover directives is not enabled. If you do not specify -asserts or -covers, the default behavior is to enable both assertions and cover directives.

- -covers

  (optional) Enables thread viewing for cover directives only. Thread viewing for assertions is not enabled. If you do not specify **-asserts** or **-covers**, the default behavior is to enable both assertions and cover directives.

- -recursive

  (optional) *For use with wildcard matching*. A switch that specifies that the scope of the matching is to descend recursively into subregions. If omitted, the search is limited to the selected region.

- <path> [<path> ...]

  (required) A space separated list of paths, that specifies the assertions to be affected. Multiple paths and wildcards are allowed. The path specifies assertions or a design region containing multiple assertions. Must be specified as the final argument to the atv log command.

Related Topics

- assertion action
- assertion active
- assertion count
- assertion enable
- assertion fail
- assertion pass
- assertion profile
- Configuring Assertions
- Viewing Assertions in the Assertions Window

batch_mode

This command returns “1” if Questa SIM is operating in batch mode, otherwise it returns “0.” It is typically used as a condition in an if statement.

Syntax

batch_mode

Arguments

None

Examples

Some GUI commands do not exist in batch mode. If you want to write a script that will work in or out of batch mode, you can use the batch_mode command to determine which command to use. For example:

```
if [batch_mode] {
   log /*
} else {
   add wave /*
}
```

Related Topics

- Modes of Operation

bd

This command deletes a breakpoint. You can delete multiple breakpoints by specifying separate information groupings on the same command line.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

bd {<filename> <line_number>}

bd {<id_number> | <label>} ...

Arguments

- <filename>

  (required when not specifying <id_number> or <label>.) A string that specifies the name of the source file in which the breakpoint is to be deleted. The filename must match the one used previously to set the breakpoint, including whether you used a full pathname or a relative name. Must be specified as the first argument to the bd command.

- <line_number>

  (required) A string that specifies the line number of the breakpoint to be deleted.

- <id_number> | <label>

  (required when not specifying <filename>.) Specifies the identification of breakpoints using markers assigned by the bp command. Must be specified as the first argument to the bd command.

  <id_number — A string that specifies the identification number of the breakpoint to be deleted. The identification number is set with the -id argument to the bp command. If you are deleting a C breakpoint, the identification number will have a "c" prefix.

  <label> — A string that specifies the label of the breakpoint to be deleted. The label is set with the -label switch to the bp command.

Examples

- Delete the breakpoint at line 127 in the source file named *alu.vhd*.

```
bd alu.vhd 127
```

- Delete the breakpoint with id# 5.

```
bd 5
```

- Delete the breakpoint with the label top_bp

```
bd top_bp
```

- Delete the breakpoint with id# 6 and the breakpoint at line 234 in the source file named *alu.vhd*.

```
bd 6 alu.vhd 234
```

- Delete the C breakpoint with id# c.4.

```
bd c.4
```

Related Topics

- bp
- onbreak
- C Debug

bookmark add wave

This command creates a named reference to a specific zoom range and scroll position in the specified Wave window. Bookmarks are saved in the wave format file and are restored when the format file is read.

You can also interactively add a bookmark through the GUI by selecting the **Wave** > **Bookmarks** > **Bookmarks** menu item.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

bookmark add wave <label> [[<range_start> [<unit>]] [<range_end> [<unit>]] [<topindex>]] [-window <window_name>]

Arguments

- <**label**>

  (required) A string that specifies the name for the bookmark. Must be specified as the first argument to the bookmark add wave command.

- <range_start> [<unit>]

  (optional) Specifies the beginning point of the zoom range where the default starting point is zero (0).

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <limit> and <unit> within curly braces ({}).

  The complete grouping of <range_start> and <range_end> must also be enclosed in braces ({ }) or quotes (" "), for example:

  ```
  {{100 ns} {10000 ns}}
  {10000}
  ```

- <range_end> [<unit>]

  (optional) Specifies the end point of the zoom range.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <limit> and <unit> within curly braces ({}).

- <topindex>

  (optional) An integer that specifies the vertical scroll position of the window. You must specify a zoom range to specify topindex. The number identifies which object the window should be scrolled to. For example, specifying 20 means the Wave window will be scrolled down to show the 20th object.

- -window <window_name>

  (optional) Specifies the window to which the bookmark will be added. If this argument is omitted, the bookmark is added in the current default Wave window.

Examples

- Add a bookmark named "foo" to the current default Wave window. The bookmark marks a zoom range from 10ns to 1000ns and a scroll position of the 20th object in the window.

```
bookmark add wave foo {{10 ns} {1000 ns}} 20
```

Related Topics

- bookmark delete wave
- bookmark goto wave
- bookmark list wave
- write format

bookmark delete wave

This command deletes bookmarks from the specified Wave window.

You can also interactively delete a bookmark through the GUI by selecting the **Wave > Bookmarks > Bookmarks** menu item.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

bookmark delete wave {<label> | -all} [-window <window_name>]

Arguments

- <label> | -all

  (required) Controls which bookmarks to delete. Must be specified as the first argument to the bookmark delete wave command.

  <label> — Specifies the name of the bookmark to delete.

  -all — Specifies that all bookmarks in the window be deleted.

- -window <window_name>

  (optional) Specifies the window from which bookmark will be deleted. If this argument is omitted, bookmark(s) in the current default Wave window are deleted.

  <window_name> — The name of the wave window containing the bookmark to be deleted.

Examples

- Delete the bookmark named "foo" from the current default Wave window.

```
bookmark delete wave foo
```

- Delete all bookmarks from the Wave window named "wave1".

```
bookmark delete wave -all -window wave1
```

Related Topics

- bookmark add wave
- bookmark goto wave
- bookmark list wave
- write format

bookmark goto wave

This command zooms and scrolls a Wave window using the specified bookmark.

You can also interactively navigate between bookmarks through the GUI by selecting the **Wave > Bookmarks > Bookmarks** menu item.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

bookmark goto wave <label> [-window <window_name>]

Arguments

- <label>

  (required) Specifies the bookmark to go to. Must be specified as the first argument to the bookmark goto wave command.

- -window <window_name>

  (optional) Specifies the Wave window to which the bookmark applies. Bookmarks can be used only in the windows in which they were originally created.

  <window_name> — The name of the wave window containing the specified bookmark.

Related Topics

- bookmark add wave
- bookmark delete wave
- bookmark list wave
- write format

bookmark list wave

This command displays a list of available bookmarks in the Transcript window.

Syntax

bookmark list wave [-window <window_name>]

Arguments

- -window <window_name>

  (optional) Specifies the Wave window to which the bookmark applies. Bookmarks can be used only in the windows in which they were originally created.

Related Topics

- bookmark add wave
- bookmark delete wave
- bookmark goto wave
- write format

bp

This command sets either a file-line breakpoint or returns a list of currently set breakpoints. It allows enum names, as well as literal values, to be used in condition expressions.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

A set breakpoint affects every SystemC instance in the design unless you use the -inst <region> argument.

Since C Debug is invoked when you set a breakpoint within a SystemC module, your C Debug settings must be in place prior to setting a breakpoint. Refer to "Setting Up C Debug" for more information.

Note

The ability to set breakpoints when running a fully optimized design may be limited. You can restrict optimizations to allow breakpoints by using the +acc=l argument to vopt. Refer to the chapter "Optimizing Designs with vopt" in the User's Manual for more information.

Syntax

Setting an HDL breakpoint

bp {[<filename>] **<line_number>** | **<filename>:<line_number>** | **in**} [-appendinst] [-cond "<condition_expression>"]
[-disable] [-id <id_number> | -label "<label>"] [-inst <region> [-inst <region> ...]] [-uvm]
[<command>…]

Setting a C breakpoint

bp **{**-c **{<function_name> | [<file_name>:]<line_number> | *0x<hex_address>}}**
[-appendinst] [-cond "<condition_expression>"] [-disable]
[-id <id_number> | -label "<label>"] [in] [-inst <region> [-inst <region> ...]]
[<command>…]

Querying a breakpoint

bp [-query <filename> [<line_number>]]

Reporting all breakpoints

If you specify this command with no arguments, it returns a list of all breakpoints in the design containing information about each breakpoint. For example:

bp

Returns:

```
# bp top.vhd 70;# 2
```

Arguments

- <filename>

  (optional) Specifies the name of the source file in which to set the breakpoint. If you do not specify a filename, the command will use the source file of the current context.

- <line_number>

  (required to set an HDL breakpoint) Specifies the line number where the breakpoint is to be set.

- in

  (required for task or function breakpoints) Supports the lookup of task and function names as an alternative to file name and line numbers. Places a breakpoint on the first executable line of the specified task or function.

- -c

  (required to set a C breakpoint) Applies the bp command and its arguments to SystemC instances in the design. Must be specified as the first argument to the **bp** command.

- <function_name> | [<file_name>:]<line_number> | *0x<hex_address>

  (required) Controls the location of a C breakpoint. Must be specified as the second argument to the bp command.

  <function_name> — Sets the C breakpoint at the entry to the specified function.

  [<file_name>:]<line_number> — Sets the C breakpoint at the specified line number of the specified file. If you do not specify a file name, the breakpoint is set at the line number of the current C or SystemC file.

  *0x<hex_address> — Sets the C breakpoint at the specified hex address.

- -appendinst

  (optional) When specifying multiple breakpoints with -inst, append each instance-path condition to the earlier condition. This overrides the default behavior, in which each condition overwrites the previous one.

- -disable

  (optional) Sets the breakpoint to a disabled state. You can enable the breakpoint later using the enablebp command. This command enables breakpoints by default.

- <command>…

  (optional, must be specified as the final argument) Specifies one or more commands that are to be executed at the breakpoint. You must separate multiple commands with semicolons (;) or place them on multiple lines. Braces are required only if the string contains spaces.

Note

You can also specify this command string by choosing **Tools > Breakpoints...** from the main menu and using the **Modify Breakpoints** dialog box.

Any commands that follow a run or step command are ignored. A run or step command terminates the breakpoint sequence. This rule also applies if you use a DO file script within the command string.

You cannot use a restore command.

If many commands are needed after the breakpoint, you could place them in a DO file script.

- -cond "<condition_expression>"

  (optional) Specifies one or more conditions that determine whether the breakpoint is hit.

  "<condition_expression>" — A conditional expression that results in a true/false value. You must enclose the condition expression within braces ({}) or quotation marks (“ ”) when the expression makes use of spaces. Refer to the note below when setting breakpoints in the GUI.

If the condition is true, the simulation stops at the breakpoint. If false, the simulation bypasses the breakpoint. A condition cannot refer to a VHDL variable (only a signal).

The -cond switch re-parses expressions each time the breakpoint is hit. This allows expressions with local references to work. Condition expressions referencing items outside the context of the breakpoint must use absolute names. This is different from the behavior in previous Questa SIM versions where a relative signal name was resolved at the time the bp command was issued, allowing the breakpoint to work even though the relative signal name was inappropriate when the breakpoint was hit.

Note

You can also specify this expression by choosing **Tools > Breakpoints...** from the main menu and entering the expression in the **Breakpoint Condition** field of the **Modify Breakpoints** dialog box. Do not enclose the condition expression in quotation marks (“ ”) or braces ({}).

The condition expression can use the following operators:

Operation	Operator Syntax
equals	==, =
not equal	!=, /=
AND	&&, AND
OR	||, OR

The operands may be object names, signame’event, or constants. Subexpressions in parentheses are permitted. The command will be executed when the expression is evaluated as TRUE or 1. The formal BNF syntax for an expression is:

```
condition ::= Name | { expression }

expression ::= expression AND relation
             | expression OR relation
```

```
                | relation
relation ::= Name = Literal
           | Name /= Literal
           | Name ' EVENT
           | ( expression )
Literal ::= '<char>' | "<bitstring>" | <bitstring>
```

The "=" operator can occur only between a Name and a Literal. This means that you cannot compare the value of two signals (for example, Name = Name is not valid).

You can construct a breakpoint such that the simulation breaks when a SystemVerilog Class is associated with a specific handle, or address:

```
bp <filename> <line_number> -cond "this==<class_handle>"
bp <filename> <line_number> -cond "this!=<class_handle>"
```

where you can obtain the class handle with the examine -handle command. The string "this" is a literal that refers to the specific *line_number*.

You can construct a breakpoint such that the simulation breaks when a line number is of a specific class type or extends the specified class type:

```
bp <filename> <line_number> -cond "this ISA <class_type_name>"
```

where *class_type_name* is the actual class name, not a variable.

- -id <id_number> | -label "<label>"

  (optional) Attempts to assign an id number or label to the breakpoint. The command returns an error if the id number you specify is already assigned.

  -id <id_number> — Any positive integer that is not already assigned.

  -label "<label>" — Associates a name or label with the specified breakpoint. Adds a level of identification to the breakpoint. The label may contain special characters. Quotation marks (" ") or braces ({ }) are required only if <label> contains spaces or special characters.

Note

Id numbers for breakpoints are assigned from the same pool as those used for the when command. So even if you have not specified a given id number for a breakpoint, that number may still be used for a when command.

- -inst <region> [-inst <region> ...]

  (optional) Sets an HDL or SystemC breakpoint so it applies only to the specified instance.

  To apply multiple instance-path conditions on a single breakpoint, specify -inst <region> multiple times. By default, this overrides the previous breakpoint condition (you can use the -appendinst argument to append conditions instead).

  <region> — The full path to the instance specified.

Note

You can also specify this instance by choosing **Tools** > **Breakpoints...** from the main menu and using the **Modify Breakpoints** dialog box.

- -query <filename> [<line_number>]

  (optional) Returns information about the breakpoint(s) set in the specified file. The information returned varies depending on the condition of the breakpoint(s) in the specified file. Returns a complete list of all breakpoints and whether they are enabled or not when specified without <line_number>. Returns nothing if <line_number> is not executable.

  <filename> — The name of the file containing the breakpoint.

  <line_number> — The line number where a breakpoint has been set.

  The output contains six fields of information. For example:

  bp -query top.vhd 70

  Returns

  ```
  # 1 1 top.vhd 70 2 1
  ```

  - {1 | 0} — Indicates whether a breakpoint exists at the location.

    0 — Breakpoint does not exit.

    1 — Breakpoint exists.
  - 1 — always reports a 1.
  - <file_name>
  - <line_number>
  - <id_number>
  - {1 | 0} — Indicates whether the breakpoint is enabled.

    0 — Breakpoint is not enabled.

    1 — Breakpoint is enabled.

- -uvm

  Specifies UVM-style instance name(s) for setting source breakpoints on class instances in the UVM hierarchy. Must be followed by the -inst <instance_name> option.

Examples

- List all existing breakpoints in the design, including the source file names, line numbers, breakpoint id#s, labels, and any commands that have been assigned to the breakpoints.

  bp

- Set a breakpoint in the source file *alu.vhd* at line 147.

bp alu.vhd 147

- Set a breakpoint at line 153 of the source file of the current context:

  bp 153

- Execute the *macro.do* DO file when the breakpoint is hit.

  bp alu.vhd 147 {do macro.do}

- Set a breakpoint on line 22 of *test.vhd*. When the breakpoint is hit, the values of variables *var1* and *var2* are examined. This breakpoint is initially disabled; it can be enabled with the enablebp command.

  bp -disable test.vhd 22 {echo [exa var1]; echo [exa var2]}

- Set a breakpoint so that the simulation pauses whenever clk=1 and prdy=0:

  bp test.vhd 14 -cond {clk=1 AND prdy=0}

- Set a breakpoint with the label top_bp

  bp top.vhd 14 -label top_bp

- Set a breakpoint for line 15 of a.vhd, but only for the instance a2:

  bp a.vhd 15 -inst "/top/a2"

- Set multiple breakpoints in the source file *test.vhd* at line 14. The second instance will overwrite the conditions of the first.

  bp test.vhd 14 -inst /test/inst1 -inst /test/inst2

- Set multiple breakpoints at line 14. The second instance will append its conditions to the first.

  bp test.vhd 14 -inst /test/inst1 -inst /test/inst2 -appendinst

- Set a breakpoint for a specific variable of a particular class type:

  set x [examine -handle my_class_var]

  bp top.sv 15 -cond {this == $x}

- Set a breakpoint on the first executable line of the function /uvm_pkg::set_config_int.

  bp in /uvm_pkg::set_config_int

- List the line number and enabled/disabled status (1 = enabled, 0 = disabled) of all breakpoints in *testadd.vhd*.

  bp -query testadd.vhd

- List details about the breakpoint on line 48.

  bp -query testadd.vhd 48

- List all executable lines in *testadd.vhd* between lines 2 and 59.

  bp -query testadd.vhd 2 59

- Set a C breakpoint at the entry to C function **and_gate_init**.

  bp -c and_gate_init

- Set a C breakpoint at line 46 in the file *and_gate.c*.

  bp -c and_gate.c:46

- Set a C breakpoint at line 44 in the current C or SystemC file.

  bp -c 44

- Set a C breakpoint at hexadecimal address **0xff130504**.

  bp -c *0xff130504

- Set a C breakpoint for instances **sctop.a.b** and **sctop.a.d**.

  bp -c -inst {sctop.a.b sctop.a.d}

- Set a C breakpoint for all instances whose name begins with **sctop.a.c**.

  bp -c -inst "sctop.a.c*"

Note

Any breakpoints set in VHDL code and called by either resolution functions or functions that appear in a port map are ignored.

Related Topics

- add button
- bd
- C Debug
- disablebp
- enablebp
- Editing File-Line Breakpoints
- onbreak
- Optimizing Designs with vopt
- SystemC Simulation
- when

call

This command calls the following types of functions/tasks.

1. SystemVerilog static functions and class functions directly from the vsim command line in live simulation mode. Tasks are not supported.
2. PLI and VPI system tasks and system functions.

Function return values are returned to the vsim shell as a Tcl string. If the function returns a class reference, the class instance ID is returned.

Syntax

Calling a System Verilog Static or Class Function

call [-env <hierEnvPath>] <pathToFunction> [<classInstancePath>] [<arg1> [<arg2>] …[<argN>]]

Calling a System Task or Function

call [-env <hierEnvPath>] [-usertf | -builtin] <systfName> [<arg1> [<arg2>] …[<argN>]]

Arguments

- <pathToFunction>

  (required when calling a System Verilog static or class function) The name of a function. The function name may be qualified in one of three ways:

  1. By specifying the path to the function declaration, through the structural hierarchy, or declaration hierarchy. Hierarchical paths must be specified as a full path to a function or a function that exists relative to the current context (as shown in the Structure window, or returned by the environment command).
  2. By specifying a class instance hierarchical path.
  3. By specifying a class instance id string.

- <systfName>

  (required when calling a system task or function) The system task or function to be executed. You can specify the sysfName according to the following syntax rules:

  4. \$<systfName> (for example, \$display or \$mytask)
  5. With –usertf or –builtin flags (as the case may be).
     (for example, –usertf mytask or –builtin display)
  6. <systfName> (for example, display or mytask) In this case SystemVerilog static functions and class functions will be searched first and then the PLI/VPI system tasks/functions, for a match.

- <arg1> [<arg2>] …[<argN>]

  (optional) All arguments required by the function are specified in a space separated list in declaration order. If a function has default arguments, the arguments may be omitted from the command line provided that the arguments occur at the end of the declaration list. Function input arguments can be constant values including integers, enumerated values, and strings. A string containing spaces or special characters must be enclosed in double quotes (" ") or braces ({ }) or Tcl will try to interpret the string. For example: "my string" or {my string}. Arguments can also be design objects. Class references can be arguments, specified by either their design instance path or class instance id string. If a function has type inout, out, or ref arguments, suitable user design objects must be passed in as arguments. Any passed in argument will first be tested to determine if it is an appropriate constant value. If it is not, then the argument will be tested to determine if it is a design object. Consequently, where there is ambiguity between a constant string and the name of a design object, the constant will be given precedence. If in this case the design object is desired, the full hierarchical path to the object can be supplied to differentiate it from the constant string.

- -builtin

  (optional) Use only built-in system functions/task names.

- <classInstancePath>

  (optional) Must be specified if the function path is a declaration path and the function is a non-static class function. Conversely, the class instance path name must not be specified if the given function path is a class instance variable reference or a class instance name in the format @<class_type>@nnn. This is because the class instance information can be extracted from the pathname itself.

- -env <hierEnvPath>

  (optional) Hierarchical environment path, to be used as the starting scope for the object name lookups. If present must appear before actual function name.

- -usertf

  (optional) Use only user-defined system functions/tasks.

Examples

Calling a System Verilog Static of Class Function

- Call using a static declaration path, where the function *sf_voidstring()* is a static class function that accepts a string:

  call sim:/user_pkg::myfcns::sf_voidstring first_string

- Call using a class instance path to specify the function, where the function *f_intint()* of the class type */utop/tmyfcns* accepts an integer:

  call /utop/tmyfcns.f_intint 37

- Call using a class instance path to specify the function, and pass in a class instance (*/utop/tmyfcns* is a class handle):

call /utop/tmyfcns.f_voidclasscolor /utop/tmyfcns

- Call using a class instance path, and pass in a class instance as an argument using a class instance id string

  call /utop/tmyfcns.f_voidclasscolor @myType@3

- Call using a class instance id string to specify the function:

  call @myType@543.get_full_name

- Call using a declaration path, where the function is non-static so a class instance must also be supplied. The member function *f_voidstring()* accepts a string:

  call sim:/user_pkg::myfcns::f_voidstring /my/class/instance "some string"

- Call using a class instance id string to specify the function where the function returns a string:

  call @uvm_sequencer__3@3.get_full_name

  Returns:

  ```
  # test.e2_a.sequencer
  ```

- Call using a relative class hierarchical name to specify the function where the function returns a class handle:

  call moduleX.who_am_i

  Returns:

  ```
  # @myClassX@4
  ```

Calling a System Task

- Call $display with literal values:

  call \$display {"%0s"} {"Hello from TCL!"}

  Returns:

  ```
  # Hello from TCL!
  ```

  call –builtin display {"%0d"} 'd2999

  Returns:

  ```
  # 2999
  ```

  call –usertf display {"%0d"} 'd3999

  Returns:

  ```
  # ** Error: Expected user-defined system task $display not found in
  the context(/top2).
  ```

- Call $display with literal values:

  call \$display {"top2.i=%0d top2.r=%0b"} top2/i top2/r

  Returns:

  ```
  # top2.i=5 top2.r=110
  ```

Calling a System Function

In the following examples $pow is a user defined function that raises the 1st argument to the power of the 2nd (for example, $pow(a, b) => a^b)

- Call $pow with literal values:

  call \$pow 2 1

  Returns:

  ```
  # 2
  ```

  call \$pow 3 2

  Returns:

  ```
  # 9
  ```

  call \$pow [call \$pow 2 1] [call \$pow 3 2]

  Returns:

  ```
  # 512
  ```

- Call $pow with variable values:

  call –env /top/u1 display r1

  Returns:

  ```
  # 7
  ```

  call –env /top/u2 display r2

  Returns:

  ```
  # 2
  ```

  call pow /top/u1/r1 /top/u2/r2

  Returns:

  ```
  # 49
  ```

cd

This command changes the Questa SIM local directory to the specified directory.

This command cannot be executed while a simulation is in progress. Also, executing a **cd** command will close the current project.

Syntax

cd [<dir>]

Arguments

- <dir>

  (optional) Specifies a full or relative directory path for QuestaSim to use as the local directory. If you do not specify a directory, the command changes to your home directory.

cdbg

This command provides command-line equivalents of the menu options that are available for C Debug.

For some of the commands there is a required argument "on | off". The value must be either “on” or “off.” For example:

```
cdbg enable_auto_step on
cdbg stop_on_quit off
```

Syntax

cdbg allow_lib_step {on | off}

cdbg auto_find_bp

cdbg debug_on

cdbg enable_auto_step {on | off}

cdbg init_mode_complete

cdbg init_mode_setup

cdbg interrupt

cdbg keep_user_init_bps {on | off}

cdbg quit

cdbg refresh_source_window

cdbg set_debugger <path>

cdbg show_source_balloon {on | off}

cdbg stop_on_quit {on | off}

cdbg trace_entry_point {on | off} [<function_name>]}

Arguments

- allow_lib_step {on | off}

  Enables stepping out from OSCI library functions. When you try to step inside OSCI library functions, C Debug automatically steps out to the last user function that was called. Note that setting this argument to “on” disables the stepping out action.

  on — Disables stepping out from OSCI library functions.

  off — Enables stepping out from OSCI library functions.

- auto_find_bp

  Sets breakpoints on all currently known function entry points.

  Equivalent to selecting **Tools** > **C Debug** > **Auto find bp**.

- **debug_on**

  Enables the C Debugger.

  Equivalent to selecting **Tools > C Debug > Start C Debug**.

- enable_auto_step {on | off}

  Specifies auto-step mode enable/disable.

  on — Enables auto-step mode.

  off — Disables auto-step mode.

  Equivalent to selecting **Tools > C Debug > Enable auto step**.

- init_mode_complete

  Instructs C Debug to continue loading the design without stopping at function calls.

  Equivalent to selecting **Tools > C Debug > Complete load**. Not supported on Windows platforms.

- init_mode_setup

  Enables initialization mode.

  Equivalent to selecting **Tools > C Debug > Init mode**. Not supported on Windows platforms.

- interrupt

  Reactivates the C debugger when stopped in HDL code.

  Equivalent to selecting **Tools > C Debug > C Interrupt** or clicking the 'C Interrupt' toolbar button.

- keep_user_init_bps {on | off}

  Specifies whether breakpoints set during initialization mode are retained after the design finishes loading.

  on — Enables retention of breakpoints after design loading is finished.

  off — Disables retention of breakpoints after design loading is finished.

  Equivalent to toggling the 'Keep user init bps' button in the C Debug setup dialog.

- quit

  Closes the C Debugger.

  Equivalent to selecting **Tools > C Debug > Quit C Debug**.

- refresh_source_window

  Re-opens a C source file if you close the Source window inadvertently while stopped in the C debugger.

  Equivalent to selecting **Tools > C Debug > Refresh**.

- set_debugger <path>

  Sets the path to your gdb installation.

  <path> — The complete pathname to the gdb executable. For example:

  cdbg set_debugger /usr/bin/gdb

  Equivalent to selecting **Tools > C Debug > C Debug Setup** and entering a custom path.

- show_source_balloon {on | off}

  Toggles the source balloon popup.

  on — Enables balloon popup.

  off — Disables balloon popup.

  Equivalent to toggling the 'Show balloon' button on the C Debug setup dialog.

- stop_on_quit {on | off}

  Toggles debugging capability when the simulator is exiting.

  on — Enables debugging when the simulator is exiting.

  off — Disables debugging when the simulator is exiting.

  Equivalent to toggling the 'Stop on quit' button on the C Debug setup dialog.

- trace_entry_point {on | off} [<function_name>]

  Helps debug an FLI/PLI application when a design is loaded with vsim -trace_foreign. Questa SIM stops at a C breakpoint each time a named FLI or PLI function is called from your application. Once at the breakpoint, use the tb and pop commands to investigate the C code at the place the function was called.

  on — Enables debugging.

  off — Disables debugging.

  <function_name> — An FLI or PLI function call.

Related Topics

- Debugging Functions During Elaboration
- Debugging Functions when Quitting Simulation
- Finding Function Entry Points with Auto Find bp
- Identifying All Registered Function Calls

change

This command modifies the value of a:

- VHDL constant, generic, or variable
- Verilog register or variable
- C variable if running C Debug

You cannot use this command on generics or parameters if you optimized the design, unless you used the vopt -floatgenerics or vopt -floatparameters arguments. These arguments allow the generics and parameters to remain floating after the optimization. Using the change command on a floating generic or parameter will change its value, but will not recompute dependent expressions. Refer to the section "Optimization of Parameters and Generics" in the User's Manual for more information.

For VHDL constants, The change command may not affect any and all uses of deferred (or other) constants. Refer to the following technote for more information:

http://supportnet.mentor.com/portal?do=reference.technote&id=MG588185

Syntax

change **<variable> <value>**

Arguments

- <variable>

  (required) A string that specifies the name of an object. The name can be a full hierarchical name or a relative name, where a relative name is relative to the current environment.

  Wildcards are not permitted.

  The following sections list supported objects:

  - VHDL
    - Scalar variable, constant, or generics of all types except FILE.

      Generates a warning when changing a VHDL constant or generic. You can suppress this warning by setting the TCL variable WarnConstantChange to 0 or in the [vsim] section of the *modelsim.ini* file.
    - Scalar subelement of composite variable, constant, and generic of all types except FILE.
    - One-dimensional array of enumerated character types, including slices.
    - Access type. An access type pointer can be set to "null"; the value that an access type points to can be changed as specified above.
  - Verilog

  - Parameter.
  - Register or memory.
  - Integer, real, realtime, time, and local variables in tasks and functions.
  - Subelements of register, integer, real, realtime, and time multi-dimensional arrays (all dimensions must be specified).
  - Bit-selects and part-selects of the above except for objects whose basic type is real.
- C
  - Scalar C variables of type int, char, double, or float.
  - Individual fields of a C structure.
  - SystemC primitive channels are not supported.

  The name can be a full hierarchical name or a relative name. A relative name is relative to the current environment. Wildcards cannot be used.

- <value>

  (required) Defines a value for <variable>. The specified value must be appropriate for the type of the variable. You must place <value> within quotation marks (“”). If the string contains spaces, the quoted string must be placed inside curly braces ({ }).

Note

The initial type of <variable> determines the type of value that it can be given. For example, if <variable> is initially equal to 3.14 then only real values can be set on it. Also note that changing the value of a parameter or generic will not modify any design elements that depended on the parameter or generic during elaboration (for example, sizes of arrays).

Examples

- Change the value of the variable *count* to the hexadecimal value FFFF.

  change count 16#FFFF

- Change the value of the element of *rega* that is specified by the index (for example, 16).

  change {rega[16]} 0

- Change the value of the set of elements of foo that is specified by the slice (for example, 20:22).

  change {foo[20:22]} 011

- Set the value of *x* (type double) to 1.5.

  change x 1.5

- Set the value of structure member *a1.c1* (type int) to 0.

change a1.c1 0

- Set *val_b* (type char *) to point to the string *my_string*.

  change val_b my_string

- Set *val_b* (type char *) to point to the string *my string*. Since there is a space in the value, it must be enclosed by curly braces.

  change val_b {“my string”}

- Set the Verilog register *file_name* to "test2.txt". Note that the quote marks are escaped with ’\’.

  change file_name \"test2.txt\"

- Set the time value of the mytimegeneric variable to 500 ps. The time value is enclosed by curly braces because of the space between the value and the units.

  change mytimegeneric {“500 ps”}

Related Topics

- force

change_menu_cmd

This command changes the command to be executed for a specified menu item label, in the specified menu, in the specified window.

The menu path and label must already exist for this command to function. Returns nothing.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

change_menu_cmd <window_name> <menu_path> <label> <cmd>

Arguments

The arguments <window_name>, <menu_path>, <cmd>, and <label> are order dependent and must be entered as shown in the Syntax section.

- <window_name>

  (required) Specifies the Tk path of the window containing the menu. The path for the Main window must be expressed as "". All other window pathnames begin with a period (.).

- <menu_path>

  (required) Specifies the name of an existing Tk menu widget plus any submenu path.

- <label>

  (required) Specifies the current label on the menu item.

- <cmd>

  (required) Specifies the new Tcl command to be executed when selected.

Related Topics

- add_menu
- add_menucb
- add_menuitem
- add_separator
- add_submenu

check contention add

This command enables contention checking for the specified nodes.

The allowed nodes are Verilog nets and VHDL signals of types std_logic and std_logic_vector. This command ignores any other node types or nodes that do not have multiple drivers.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

check contention add {[-in] [-out] [-inout] | [-ports]} [-internal] [-r] <node_name>...

Arguments

- -in

  (optional) Enables checking on nodes of mode IN.

- -inout

  (optional) Enables checking on nodes of mode INOUT.

- -out

  (optional) Enables checking on nodes of mode OUT.

- -ports

  (optional) Enables checking on nodes of modes IN, OUT, or INOUT. Default behavior if no arguments are specified.

- -internal

  (optional) Enables checking on internal (non-port) objects. Default behavior if no arguments are specified.

- -r

  (optional) Specifies that contention checking is enabled recursively into subregions. If omitted, contention check enabling is limited to the current region.

- <node_name>...

  (required) Specifies the name of a node. Must be specified as the final argument to the check contention add command.

Description

Bus contention checking detects bus fights on nodes that have multiple drivers. A bus fight occurs when two or more drivers drive a node with the same strength and that strength is the strongest of all drivers currently driving the node. The following table provides some examples for two drivers driving a std_logic signal:

driver 1	driver 2	fight
Z	Z	no

driver 1	driver 2	fight
0	0	yes
1	Z	no
0	1	yes
L	1	no
L	H	yes

Detection of a bus fight results in an error message specifying the node and its drivers' current driving values. If a node's drivers later change value and the node is still in contention, a message is issued giving the new values of the drivers. A message is also issued when the contention ends. The bus contention checking commands can be used on VHDL and Verilog designs.

Related Topics

- check contention config
- check contention off

check contention config

This command allows you to write checking messages to a file. By default, any messages display on your screen.

You may also configure the contention time limit.

Syntax

check contention config [-file <filename>] [-time <limit>[<unit>]]

Arguments

- -file <filename>

  (optional) Specifies to write contention messages to a specified file. When you specify this switch, check contention messages will not be displayed to the screen.

  <filename> — The name of a file where contention messages will be saved.

- -time <limit>[<unit>]

  (optional) Specifies a time limit that a node may be in contention. Contention is detected if a node is in contention for as long as or longer than the limit.

  <limit> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <limit> and <unit> within curly braces ({}).

Related Topics

- check contention add
- check contention off

check contention off

This command disables contention checking for the specified nodes.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

check contention off [-all] {[-in] [-out] [-inout] | [-ports]} [-internal] [-r] <node_name> ...

Arguments

- -all

  (optional) Disables contention checking for all nodes that have checking enabled.
- -in

  (optional) Disables checking on nodes of mode IN.
- -out

  (optional) Disables checking on nodes of mode OUT.
- -inout

  (optional) Disables checking on nodes of mode INOUT.
- -ports

  (optional) Disables checking on nodes of modes IN, OUT, or INOUT.
- -internal

  (optional) Disables checking on internal (non-port) objects.
- -r

  (optional) Specifies that contention checking is disabled recursively into subregions. If omitted, contention check disabling is limited to the current region.
- <node_name> ...

  (required) Specifies the named node(s). Must be specified as the final argument to the check contention off command.

Related Topics

- check contention add
- check contention config

check float add

This command enables float checking for the specified nodes.

The allowed nodes are Verilog nets and VHDL signals of type std_logic and std_logic_vector (other types are silently ignored).

You can set a time limit (the default is zero) for float checking using the -time <limit> argument to the check float config command. If you choose to modify the limit, you should do so prior to invoking any check float add commands.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

check float add {[-in] [-out] [-inout] | [-ports]} [-internal] [-r] <node_name>...

Arguments

- -r

  (optional) Specifies that float checking is enabled recursively into subregions. If omitted, float check enabling is limited to the current region.

- -in

  (optional) Enables checking on nodes of mode IN.

- -out

  (optional) Enables checking on nodes of mode OUT.

- -inout

  (optional) Enables checking on nodes of mode INOUT.

- -internal

  (optional) Enables checking on internal (non-port) objects.

- -ports

  (optional) Enables checking on nodes of modes IN, OUT, or INOUT.

- <node_name>...

  (required) Enables checking for the named node(s). Must be specified as the final argument to the check float add command.

Description

Bus float checking detects nodes that are in the high impedance state for a time equal to or exceeding a user-defined limit. This is an error in some technologies. Detection of a float violation results in an error message identifying the node. A message is also issued when the float violation ends. The bus float checking commands can be used on VHDL and Verilog designs.

Related Topics

- check contention config
- check float off

check float config

This command allows you to write checking messages to a file (messages display on your screen by default). You may also configure the float time limit.

Syntax

check float config [-file <filename>] [-time <time><unit>]

Arguments

- -file <filename>

  (optional) Writes float messages to a file. If this option is selected, the messages are not displayed to the screen.

  <filename — Specifies the name of the file that float messages will be saved to.

- -time <time><unit>

  (optional) Specifies a length of time that a node may be floating. An error is detected if a node is floating for as long as or longer than the limit. Note that you should configure the time limit prior to invoking any check float add commands.

  <time> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

Related Topics

- check float add
- check float off

check float off

This command disables float checking for the specified nodes.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

check float off [-all] {[-in] [-out] [-inout] | [-ports]} [-internal] [-r] <node_name> ...

Arguments

- -all

  (optional) Disables float checking for all nodes that have checking enabled.
- -in

  (optional) Disables checking on nodes of mode IN.
- -out

  (optional) Disables checking on nodes of mode OUT.
- -inout

  (optional) Disables checking on nodes of mode INOUT.
- -ports

  (optional) Disables checking on nodes of modes IN, OUT, or INOUT.
- -internal

  (optional) Disables checking on internal (non-port) objects.
- -r

  (optional) Specifies that float checking is disabled recursively into subregions. If omitted, float check disabling is limited to the current region.
- <node_name> ...

  (required) Disables checking for the named node(s). Must be specified as the final argument to the check float off command.

Related Topics

- check float add
- check float config

check stable off

This command disables stability checking.

You may later enable it with check stable on, and meanwhile, the clock cycle numbers and boundaries are still tracked.

Syntax

check stable off

Arguments

None

Related Topics

- check stable on

check stable on

This command enables stability checking on the entire design.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

check stable on **[-period <time> [<unit>]]** [-file <filename>] [-strobe <time> [<unit>]]

Arguments

- -period <time> [<unit>]

  (required the first time you invoke the check stable on command.) Specifies the clock period (which is assumed to begin at the time the check stable on command is issued). Must be specified as the first argument to the check stable on command.

  <time> — Any non-negative integer.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

  This argument is not required if you later enable checking after it was disabled with the check stable off command.

- -file <filename>

  (optional) Writes the error messages to a specified file. If this option is selected, the messages are not displayed to the screen.

  <filename> — Specifies the name of the file that error messages will be saved to.

- -strobe <time> [<unit>]

  (optional) Specifies the elapsed time within each clock cycle that the stability check is performed. The default strobe time is the time specified in -period. If the strobe time falls on a period boundary, then the check is actually performed one timestep earlier. Normally the strobe time is specified as less than or equal to -period, but if it is greater than -period, then the check will skip cycles.

  <time> — Any non-negative integer.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

Description

Design stability checking detects when circuit activity has not settled within a period you define for synchronous designs. You specify the clock period for the design and the strobe time within the period during which the circuit must be stable. A violation is detected and an error message

is issued if there are pending driver events at the strobe time. The message identifies the driver that has a pending event, the node that it drives, and the cycle number. The design stability checking commands can be used on VHDL and Verilog designs.

Examples

- Performs a stability check 99 ps into each even numbered clock cycle (cycle numbers start at 1).

```
check stable on -period "100 ps" -strobe "199 ps"
```

Related Topics

- check stable off

checkpoint

This command saves the state of your simulation, including:

- *modelsim.ini* settings.
- the simulation kernel state.
- the *vsim.wlf* file.
- the list of the design objects shown in the List and Wave windows.
- the file pointer positions for files opened under VHDL and the Verilog $fopen system task.
- the states of foreign architectures.
- VCD output.
- Toggle statistics (see the toggle report command).

Once saved, you can use a checkpoint file with the restore command during the same simulation to restore the simulation to a previous state. You can also start a vsim session with a checkpoint file by using the vsim -restore command.

Compression of the checkpoint file is controlled by the CheckpointCompressMode variable in the *modelsim.ini* file.

If a checkpoint occurs while Questa SIM is writing a VCD file, the entire VCD file is copied into the checkpoint file. Because VCD files can be very large, it is possible that disk space problems could occur. Consequently, Questa SIM issues a warning in this situation.

If checkpointing DPI code that works with heap memory, use mti_Malloc() rather than raw malloc() or new. Any memory allocated with mti_Malloc() is guaranteed to be restored correctly. Any memory allocated with raw malloc() will not be restored correctly, and simulator crashes can result.

Restrictions

- Checkpoint files are platform dependent—you cannot checkpoint on one platform and restore on another.
- Changes you made interactively while running vsim are not saved. For example, macros, virtual objects, command-line interface additions such as user-defined commands, and states of graphical user interface are not saved.
- Transactions are not saved.
- The checkpoint command is not supported once a SystemC design has been loaded.

Syntax

checkpoint [-dir] <pathname>

Arguments

- -dir

  (optional) Creates a directory with the specified pathname. The following are the two valid scenarios for specifying a directory with the <pathname>.

  - The "-dir" option is specified with the <pathname>. If the directory doesn't exist then a new one is created.
  - The "-dir" option is not specified with the <pathname> but the <pathname> represents a pre-exisiting directory.

  In both these cases a checkpoint file named '*vsim.cpt*' is created in the specified directory.

- <pathname>

  (required) Specifies the pathname of the directory where the *vsim.cpt* checkpoint file is saved.

Related Topics

- restore
- restart
- vsim
- Checkpointing and Restoring Simulations

classinfo ancestry

This command returns class inheritance hierarchy for a named class type.

Syntax

classinfo ancestry [-dataset <name>] [-n] [-tcl] [-o <outfile>] <class_type>

Arguments

- <class_type>

  (required) Name of the class type or the full path to the class type.

- -dataset <name>

  (optional) Specifies an open dataset to search for class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -n

  (optional) Returns class type names only. Does not include the path unless required to resolve name ambiguity.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

- -tcl

  (optional) Returns a tcl list instead of formatted output.

Related Topics

- ClassDebug modelsim.ini variable
- vsim -classdebug
- classinfo descriptive
- classinfo find
- classinfo instances
- classinfo isa
- classinfo report
- classinfo stats
- classinfo trace
- classinfo types

classinfo descriptive

This command returns the descriptive class name for the specified authoritative class name.

Syntax

classinfo descriptive [-dataset <name>] [-exact | -glob | -regexp] [-n] [-tcl] [-o <outfile>] <class_type>

Arguments

- <class_type>

  (required) Treats <class_type> as a glob-style expression and returns all matches to the transcript. Wildcard characters asterisk (*) and question mark (?) are permitted.

- -dataset <name>

  (optional) Specifies an open dataset to search for class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -exact

  (optional) Returns results that match <class_type> exactly.

- -glob

  (optional, default) Treats <class_type> as a glob-style expression. Wildcard characters asterisk (*) and question mark (?) are permitted.

- -n

  (optional) Returns class names only. Does not include the path unless required to resolve name ambiguity.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

- -regexp

  (optional) Treats <class_name> as a regular expression.

- -tcl

  (optional) Returns a tcl list instead of formatted output.

Related Topics

- ClassDebug modelsim.ini variable
- Logging Class Types and Class Instances
- Working with Class Types
- Analyzing Class Types
- vsim -classdebug
- classinfo ancestry
- classinfo find
- classinfo instances
- classinfo isa
- classinfo report
- classinfo stats
- classinfo trace
- classinfo types

classinfo find

This command reports on the current state of a specified class instance, whether it exists, has not yet been created, or has been destroyed.

Syntax

classinfo find [-dataset <name>] [-tcl] [-o <outfile>] **<class_instance_identifier>**

Arguments

- <class_instance_identifier>

  (required) Class instance identifier of the specific class instance to find.

- -dataset <name>

  (optional) Specifies an open dataset to search for class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

- -tcl

  (optional) Returns a tcl list instead of formatted output.

Examples

- Find the class instance @mem_item@87

  VSIM> classinfo find @mem_item@87

  Returns:

  ```
  # @mem_item@87 has been destroyed
  ```

- Find the class instance @mem_item@200

  VSIM> classinfo find @mem_item@200

  Returns:

  ```
  # @mem_item@200 not yet created
  ```

Related Topics

- ClassDebug modelsim.ini variable
- vsim -classdebug
- classinfo ancestry
- classinfo descriptive
- classinfo instances
- classinfo isa
- classinfo report
- classinfo stats
- classinfo trace
- classinfo types

classinfo implements

This command displays a list of which classes implement SystemVerilog interface classes. The type of the class argument affect the contents of this list.

Syntax

classinfo implements [-dataset <name>] [-n] [-tcl] [-o <outfile>] <class_type>

Arguments

- <class>

  (required) Name of the SystemVerilog class that you want to use to generate the output listing. The type of this class determines the type of classes listed, as follows:

  - If <class_type> is not an interface class, the output indicates which interface classes that class implements.
  - If <class_type> is an interface class, the output indicates which classes implement that interface class.

- -dataset <name>

  (optional) Specifies an open dataset to search for class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -n

  (optional) Returns class type names only. Does not include the path unless required to resolve name ambiguity.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

- -tcl

  (optional) Returns a tcl list instead of formatted output.

Examples

The following module defines classes labeled A, B, C1, C2, M, N, X, Y:

```
module test8;
  interface class M;  endclass
  interface class N;  endclass
  interface class X extends M, N; endclass
  interface class Y extends M;  endclass

  class A implements M; endclass
  class B extends A implements X; endclass
  class C1 extends B implements Y; endclass
  class C2 extends B; endclass
```

```
endmodule
```

- Use interface class M as argument:

  vsim> classinfo implements M

  Output list:

  ```
  # /test8/A implements /test8/M
  # /test8/B implements /test8/M
  # /test8/C1 implements /test8/M
  # /test8/C2 implements /test8/M
  ```

- Use class A as argument:

  vsim> classinfo implements A

  Output list:

  ```
  # /test8/A implements /test8/M
  ```

- Use class B as argument to access extended classes defined in test8:

  vsim> classinfo implements B

  Output list:

  ```
  # /test8/B implements /test8/M
  # /test8/B implements /test8/N
  # /test8/B implements /test8/X
  ```

Related Topics

- ClassDebug modelsim.ini variable
- vsim -classdebug
- classinfo descriptive
- classinfo find
- classinfo instances
- classinfo interfaces
- classinfo isa
- classinfo report
- classinfo stats
- classinfo trace
- classinfo types

classinfo instances

This command reports the list of existing class instances of a specific class type. You can use this to determine what class instances to log or examine. It can also help in debugging problems where class instances are not being cleaned up as they should be resulting in excessive memory usage.

Syntax

classinfo instances [-dataset <name>] [-tcl] [-verbose] [-o <outfile>] **<class_type>**

Arguments

- <class_type>

  (required) Name of the class type or the full path of the class type. If this is an interface class, the output lists all instances that implement that interface class.

- -dataset <name>

  (optional) Specifies an open dataset to search for class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -tcl

  (optional) Returns a tcl list instead of formatted output.

- -verbose

  (optional) Includes the classname in the output along with the instance name.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

Examples

- List the current instances for the class type mem_item.

  vsim> classinfo instances mem_item

  Returns:

  ```
  # @mem_item@140
  # @mem_item@139
  # @mem_item@138
  # @mem_item@80
  # @mem_item@76
  # @mem_item@72
  # @mem_item@68
  # @mem_item@64
  ```

- The following module defines classes labeled A, B, C1, C2, M, N, X, Y:

```
module test8;
  interface class M;  endclass
  interface class N;  endclass
  interface class X extends M, N; endclass
  interface class Y extends M;  endclass

  class A implements M; endclass
  class B extends A implements X; endclass
  class C1 extends B implements Y; endclass
  class C2 extends B; endclass
endmodule
```

The following commands show the difference between using and omitting the -verbose argument.

vsim> classinfo instances -verbose M

Returns:

```
# @A@1 /test8/A
# @B@1 /test8/B
```

vsim> classinfo instances -verbose A

Returns:

```
# @A@1 /test8/A
```

vsim> classinfo instances M

Returns:

```
# @A@1
# @B@1
```

vsim> classinfo instances A

Returns:

```
# @A@1
```

Related Topics

- ClassDebug modelsim.ini variable
- vsim -classdebug
- classinfo ancestry
- classinfo descriptive
- classinfo find
- classinfo implements
- classinfo interfaces
- classinfo isa
- classinfo report
- classinfo stats
- classinfo trace
- classinfo types

classinfo interfaces

This command lists the interface class types that match or do not match a specified pattern. Finds all interface classes that match a regular expression and determines the full path of interface class types.

Syntax

classinfo interfaces [-dataset <name>] [-tcl] [-o <outfile>] [<class_type>]

Arguments

- <class_type>

  (optional) Name of the interface class type or the full path to the interface class type. If omitted, all interface classes are listed.

- -dataset <name>

  (optional) Specifies an open dataset to search for interface class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

- -tcl

  (optional) Returns a tcl list instead of formatted output.

Examples

The following module defines classes labeled A, B, C1, C2, M, N, X, Y:

```
module test8;
  interface class M;  endclass
  interface class N;  endclass
  interface class X extends M, N; endclass
  interface class Y extends M;  endclass

  class A implements M; endclass
  class B extends A implements X; endclass
  class C1 extends B implements Y; endclass
  class C2 extends B; endclass
endmodule
```

- Use with no argument, which will return the names of all interface classes:

  vsim> classinfo interfaces

  Output list:

  ```
  # /test8/M
  ```

```
# /test8/N
# /test8/X
# /test8/Y
```

Related Topics

- ClassDebug modelsim.ini variable
- vsim -classdebug
- classinfo descriptive
- classinfo find
- classinfo implements
- classinfo instances
- classinfo isa
- classinfo report
- classinfo stats
- classinfo trace
- classinfo types

classinfo isa

This command returns to the transcript a list of all classes extended from the specified class type.

Syntax

classinfo isa [-dataset <name>] [-n] [-o <outfile>] [-tcl] **<class_type>**

Arguments

- <class_type>

  (required) Name of the class type or the full path of the class type.

- -dataset <name>

  (optional) Specifies an open dataset to search for class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -n

  (optional) Returns class names only. Does not include the path unless required to resolve name ambiguity.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

- -tcl

  (optional) Returns a tcl list instead of formatted output.

Related Topics

- ClassDebug modelsim.ini variable
- vsim -classdebug
- classinfo ancestry
- classinfo descriptive
- classinfo find
- classinfo instances
- classinfo report
- classinfo stats
- classinfo trace
- classinfo types

classinfo report

This command prints detailed reports on class instance usage. The command displays columns for class type names and their current, peak and total class instance counts. The columns may be arranged, sorted, or eliminated using the command arguments.

Syntax

classinfo report [-c [fntpc]] [-dataset <name>] [-m <maxout>] [-o <outfile>] [-sort [a | d][f | n | t | p | c]] [-tcl] [-z]

Arguments

- -c [fntpc]

  (optional) Display the report columns in the specified order in a report. The default is to display all columns in the following order: Full Path, Class Name, Total, Peak, Current. You can specify one or more columns in any order.

  f — The Full Path column displays the full relative path name.

  n — The Class Name column displays the name of the class instance.

  t — The Total column displays the total number of instances of the named class.

  p — The Peak column displays the maximum number of instances of the named class that existed simultanously at any time in the simulation.

  c — The Current column displays the current number of instances of the named class.

- -dataset <name>

  (optional) Specifies an open dataset to search for class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -m <maxout>

  (optional) Display the specified number of lines of the report.

  <maxout> — Any non-negative integer.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

- -sort [a | d][f | n | t | p | c]

  (optional) Specifies whether the report information is sorted in ascending or descending order and which column to sort by. Only one column can be specified for sorting.

  a — Sort the entries in ascending order.

  d — Sort the entries in descending order.

  f — Sort by the Full Path column

n— Sort by the Class Name column

t — Sort by the Total column

p — Sort by the Peak column

c — Sort by the Current column

- -tcl

  (optional) Returns a tcl list instead of formatted output.

- -z

  (optional) Remove all items from the report with a total instance count of zero.

Examples

- Create a report of all class instances in descending order in the Total column. Print the Class Names, Total, Peak, and Current columns. List only the first six lines of that report.

  vsim> classinfo report -s dt -c ntpc -m 6

  Returns:

```
# Class Name                Total   Peak Current
# uvm_pool__11                318    315     315
# uvm_event                   286     55      52
# uvm_callback_iter__1        273      3       2
# uvm_queue__3                197     13      10
# uvm_object_string_pool__1   175     60      58
# mem_item                    140     25      23
```

Related Topics

- ClassDebug modelsim.ini variable
- vsim -classdebug
- classinfo ancestry
- classinfo descriptive
- classinfo find
- classinfo instances
- classinfo isa
- classinfo stats
- classinfo trace
- classinfo types

classinfo stats

This command prints statistics about the total number of class types and total, peak, and current class instance counts during the simulation.

Syntax

classinfo stats [-dataset <name>] [-tcl] [-o <outfile>]

Arguments

- -dataset <name>

  (optional) Specifies an open dataset to search for class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

- -tcl

  (optional) Returns a tcl list instead of formatted output.

Examples

- Display the current number of class types, the maximum number, peak number and current number of all class instances.

  vsim> classinfo stats

  Returns:

```
# class type count                451
# class instance count (total)   2070
# class instance count (peak)    1075
# class instance count (current) 1058
```

Related Topics

- ClassDebug modelsim.ini variable
- vsim -classdebug
- classinfo ancestry
- classinfo descriptive
- classinfo find
- classinfo instances
- classinfo isa
- classinfo report
- classinfo trace
- classinfo types

classinfo trace

This command displays the active references to the specified class instance. This is very useful in debugging situations where class instances are not being destroyed as expected because something in the design is still referencing them. Finding those references may lead to uncovering bugs in managing these class references which often lead to large memory savings.

Syntax

classinfo trace [-dataset <name>] [-m <maxout>] [-tcl] [-o <outfile>] **<class_instance_name>**

Arguments

- <class_instance_name>

  (required) Name of the class item in the following format @<name>@#.

- -dataset <name>

  (optional) Specifies an open dataset to search for class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -m <maxout>

  (optional) Display the specified number of lines of the report.

  <maxout> — Any non-negative integer.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

- -tcl

  (optional) Returns a tcl list instead of formatted output.

Examples

- Report the active references to @mem_item@200

  VSIM> classinfo trace @uvm_resource__14@2

  Returns:

```
#{uvm_pkg::uvm_resources.rtab["mem_interface"].queue[15]}
#{uvm_pkg::uvm_config_db::uvm_config_db__12::m_rsc[@uvm_root@1].
pool["uvm_test_topmem_interface"]}
```

Related Topics

- ClassDebug modelsim.ini variable
- vsim -classdebug
- classinfo ancestry
- classinfo descriptive
- classinfo find
- classinfo instances
- classinfo isa
- classinfo report
- classinfo stats
- classinfo types

classinfo types

This command lists the class types that match or do not match a specified pattern. Finds all classes that match a regular expression and determines the full path of class types.

Syntax

classinfo types [-dataset <name>] [-exact | -glob | -regexp] [-n] [-o <outfile>] [-tcl] [-x] **<pattern>**

Arguments

- <pattern>

  (required) A standard TCL glob expression used as a search string.

- -dataset <name>

  (optional) Specifies an open dataset to search for class information. The default is to search the currently active dataset.

  <name> — The name of an open dataset.

- -exact

  (optional) Returns results that match <pattern> exactly.

- -glob

  (optional) Returns glob styles matches for <pattern>.

- -n

  (optional) Returns class names only. Does not include the path unless required to resolve name ambiguity.

- -o <outfile>

  (optional) Sends the results of the command to <outfile> instead of the transcript.

  <outfile> — Specifies the name of the file where the output will be written.

- -regexp

  (optional) Returns regular expressions that match <pattern>.

- -tcl

  (optional) Returns a tcl list instead of formatted output.

- -x

  (optional) Display classes that do not match the pattern.

Examples

- List the full path of the class types that do not match the pattern *uvm*.

```
vsim> classinfo types -x *uvm*
```

Returns:

```
# /environment_pkg::test_predictor
# /environment_pkg::threaded_scoreboard
# /mem_agent_pkg::mem_agent
# /mem_agent_pkg::mem_config
# /mem_agent_pkg::mem_driver
```

Related Topics

- ClassDebug modelsim.ini variable
- vsim -classdebug
- classinfo ancestry
- classinfo descriptive
- classinfo find
- classinfo implements
- classinfo instances
- classinfo interfaces
- classinfo isa
- classinfo report
- classinfo stats
- classinfo trace

compare add

This command creates an object that is a comparison between signals in a reference design against signals in a test design. You can specify whether to compare two signals, all signals in the region, just ports or a subset of ports. Constant signals such as parameters and generics are ignored.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Refer to "Waveform Compare" for a general overview of waveform comparisons.

The names of the added comparison objects take the form:

<path>/\refSignalName<>testSignalName

If you compare two signals from different regions, the signal names include the uncommon part of the path. Table 2-3 shows how comparisons work between specified reference objects and test objects.

Table 2-3. Comparing Reference Objects to Test Objects

Reference object	Test object	Result
signal	signal	compare the two signals
signal	region	compare a signal with a name matching the reference signal in the specified test region
region	region	compare all matching signals in both regions
glob expression	signal	legal only if the glob expression selects only one signal
glob expression	region	compare all signals matching the glob expression that match signals in the test region

The compare add command supports arguments that specify how each signal state matches std_logic or Verilog values (for example, -vhdlmatches, see below). Since state matching can also be set on a global basis with the compare options command or PrefCompare() Tcl variables, Questa SIM follows state match settings in the following order:

1. Use local matching values specified when the comparison was created using compare add or subsequently configured using compare configure.
2. If no local values were set, use global matching values set with the compare options command.
3. If no compare options were set, use default matching values specified by PrefCompare Tcl variables.

Syntax

compare add **[**-all**]** {[-in] [-inout] [-out] | [-ports]} [-internal] [-label <label>] [-list] [-rebuild] [-recursive] [-separator <string>] [<testPath>] [-tol <time> [<unit>]] [-tolLead <time> [<unit>]] [-tolTrail <time> [<unit>]] [-verbose] [-vhdlmatches {<ref-logic-value>=<test-logic-value>:…}] [-vlogmatches {<ref-logic-value>=<test-logic-value>:…}] [-wave] [-wavepane <n>] [-when {"<expression>"}] [-win <wname>] [-nowin] {**-clock <name>} <referencePath>**

Arguments

- -all

  (optional) Specifies comparison of all signals.

- -help

  (optional) Lists the description and syntax for the compare add command in the Transcript window.

- -in

  (optional) Specifies comparison of ports of mode IN. Multiple modes can be specified in the same command.

- -inout

  (optional) Specifies comparison of ports of mode INOUT. Multiple modes can be specified in the same command.

- -out

  (optional) Specifies comparison of ports of mode OUT. Multiple modes can be specified in the same command.

- -ports

  (optional) Specifies comparison of all ports. This switch has the same effect as specifying -in, -out, and -inout together.

- -internal

  (optional) Specifies comparison of internal (non-port) signals. Multiple modes can be specified in the same command.

- -label <label>

  (optional) Specifies a name for the comparison when it is displayed in the Wave window.

  <label> — The specified name for the comparison.

- -list

  (optional) Causes specified comparisons to be displayed in the default List window.

- -rebuild

  (optional) Rebuilds a fragmented bus in the test design region and compares it with the corresponding bus in the reference design region. If a signal is found having the same name

as the reference signal, the -rebuild option is ignored. When rebuilding the test signal, the name of the reference signal is used as the wildcard prefix.

- -recursive

  (optional) Specifies that signals should also be selected in all nested subregions, and subregions of those, and so forth.

- -separator <string>

  (optional) Used with the -rebuild argument. Specifies a separator to be inserted between a base bus name and the bit indication. When a bus has been broken into bits (bit blasted) by a synthesis tool, Questa SIM expects a separator between the base bus name and the bit indication. For example, the signal "mybus" might be broken down into "mybus_0", "mybus_1", and so forth.

  <string> — Specifies the character(s) to be used as a separator between the base bus name and the bit indication where the default is an underline “_”.

  For example, the signal "mybus" might be broken down into "mybus_0", "mybus_1", and so forth.

- <testPath>

  (optional) Specifies an absolute or relative path to the test signal or region. Cannot be a glob expression. If omitted, the test path defaults to the same path as <referencePath> except for the dataset name.

- -tol <time> [<unit>]

  (optional) Specifies the maximum amount of time a test signal edge is allowed to lead or trail a reference edge in an asynchronous comparison.

  <time> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -tolLead <time> [<unit>]

  (optional) Specifies the maximum amount of time a test signal edge is allowed to lead a reference edge in an asynchronous comparison.

  <time> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -tolTrail <time> [<unit>]

  (optional) Specifies the maximum amount of time a test signal edge is allowed to trail a reference edge in an asynchronous comparison.

<time> — Any non-negative integer where the default is 0.

<unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

Graphical representation of tolLead and tolTrail

- -verbose

  (optional) Prints information in the Transcript window confirming the signals selected for comparison and any type conversions employed.

- -vhdlmatches {<ref-logic-value>=<test-logic-value>:…}

  (optional) Specifies how VHDL signal states in the reference dataset should match values in the test dataset. Values are specified in a colon-separated list of match values. For example:

  -vhdlmatches {X=XUD:Z=ZD:1=1HD}

  Default is:

  ```
  {U=UWXD:X=UWXD:0=0LD:1=1HD:Z=ZD:W=UWXD:L=0LD:H=1HD:D=UX01ZWLHD}
  ```

  The 'D' character represents the '-' "don't care" std_logic value.

- -vlogmatches {<ref-logic-value>=<test-logic-value>:…}

  (optional) Specifies how Verilog signal states in the reference dataset should match values in the test dataset. Values are specified in a colon-separated list of match values. For example:

  -vlogmatches {0=0:1=1:Z=Z}

  Default is:

  ```
  {0=0:1=1:Z=Z:X=X}.
  ```

- -wave

  (optional) Specifies that compare signals be added automatically to the default Wave window. (default)

- -wavepane <n>

  (optional) Specifies the pane of the Wave window in which the differences will be placed.

<n> — A positive integer corresponding to the pane of the Wave window. Wave window panes are numbered in sequence starting with 1 for the initial wave window.

- -when {“<expression>”}

  (optional) Specifies a conditional expression that must evaluate to "true" or "1" for differences to be reported. The expression is evaluated at the start of an observed difference. See GUI_expression_format for legal expression syntax.

  "<expression>" — A conditional expression that results in a true/false value. You must enclose the condition expression within quotation marks (“ ”).

- -win <wname>

  (optional) Adds objects to a specified window. Used to specify a particular window when multiple instances of that window type exist.

  <wname> — The name of the window to which objects are to be added.

- -nowin

  (optional) Specifies that compare signals shouldn’t be added to any window, where the default is to add compare signals to the default Wave window. See -wave.

- -clock <name>

  (required for clocked comparisons; not used for asynchronous comparisons.) Specifies the clock definition to use when sampling the specified regions.

  <name> — Specifies the name of the clock definition.

- <referencePath>

  (required) Specifies either an absolute or relative path to the reference signal or region, or a glob expression. Relative paths are relative to the current context of the reference dataset. If you specify a glob expression, it will match signals only in the containing context. Must be specified as the final argument to the compare add command.

Examples

- Select signals in the reference and test dataset top region according to the default mode. Uses asynchronous comparison with the default tolerances. Assumes that the top regions of the reference and test datasets have the same name and contain the same signals with the same names.

  compare add /*

- Select port signals of instance *.test_ringbuf.ring_inst* in both datasets to be compared and sampled on strobe *myclock10*.

  compare add -port -clock myclock10 gold:.test_ringbuf.ring_inst

- Select all signals in the *cpu* region to be compared asynchronously using the default tolerances. Requires that the reference and test relative hierarchies and signal names within the *cpu* region be identical, but they need not be the same above the *cpu* region.

  compare add -r gold:/top/cpu test:/testbench/cpu

- Specify that signal *gold:.top.s1* should be sampled at *clock12* and compared with *test:.top.s1*, also sampled at *clock12*.

  compare add -clock clock12 gold:.top.s1

- Specify that signal *gold:/asynch/abc/s1* should be compared asynchronously with signal *sim:/flat/sigabc* using a leading tolerance of 3 ns and a trailing tolerance of 5 ns.

  compare add -tolLead {3 ns} -tolTrail {5 ns} gold:/asynch/abc/s1 sim:/flat/sigabc

- Cause signals *test:.counter2.cnt_dd* to be rebuilt into bus *test:.counter2.cnt[…]* and compared against *gold:.counter1.count*.

  compare add -rebuild gold:.counter1.count test:.counter2.cnt

Related Topics

- compare add
- compare annotate
- compare clock
- compare configure
- compare continue
- compare delete
- compare end
- compare info
- compare list
- compare options
- compare reload
- compare reset
- compare run
- compare savediffs
- compare saverules
- compare see
- compare start
- compare stop
- compare update
- Waveform Compare

compare annotate

This command either flags a comparison difference as "ignore" or adds a text string annotation to the difference. The text string appears when the difference is viewed in info popups or in the output of a compare info command.

Some of the arguments for this command are order-dependent. Please read through the arguments for further information.

Syntax

compare annotate [-ignore] [-noignore] [-text <message>] <idNum1> <idNum2>…]

Arguments

- -ignore

  (optional) Flags the specified difference as "ignore."

- -noignore

  (optional) Undoes a previous -ignore argument.

- -text <message>

  (optional) Adds a text string annotation to the difference that is shown wherever the difference is viewed.

- <idNum1> <idNum2>…

  (required) Identifies the difference number to annotate. You can obtain a difference number by using the compare start command or by placing the cursor over the logged difference in the Wave window and reading the popup dialog. Difference numbers are ordered by time of the difference start, but there may be more than one difference starting at a given time. Multiple idNumbers are specified as a space separated list. Must be specified as the final argument to compare annotate.

  If you specify this argument with no other arguments, it returns the current annotations recorded for the specified IDs. For example:

  compare annotate 40

  returns:

  ```
  # Diff 40: -ignore -text "This is a not a critical problem."
  ```

  - Diff 40— an echo of the specified comparison difference.
  - The current setting of -ignore or -noignore.
  - -text “<text string saved to the specified difference>” If no text has been saved to the difference, an empty set of quotation marks (““) is returned.

Examples

- Flag difference numbers 1, 2, and 10 as "ignore."

compare annotate -ignore 1 2 10

- Annotate difference number 12 with the message "THIS IS A CRITICAL PROBLEM."

  compare annotate -text "THIS IS A CRITICAL PROBLEM" 12

Related Topics

- compare add
- compare info
- Waveform Compare

compare clock

This command defines a clock that can then be used for clocked-mode comparisons. In clocked-mode comparisons, signals are sampled and compared only at or just after an edge on some signal.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

compare clock [-delete] [-offset <delay>[<unit>]] [-rising | -falling | -both] [-when {“<expression>”}}] {<clock_name> <signal_path>}

Arguments

- -delete

  (optional) Deletes an existing compare clock.

- -offset <delay>[<unit>]

  (optional) Specifies a time value for delaying the sample time beyond the specified signal edge.

  <delay> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <delay> and <unit> within curly braces ({}).

- -rising

  (optional) Specifies that the rising edge of the specified signal should be used. (default)

- -falling

  (optional) Specifies that the falling edge of the specified signal should be used. The default is rising.

- -both

  (optional) Specifies that both the rising and the falling edge of the specified signal should be used. The default is rising.

- -when {“<expression>”}

  (optional) Specifies a conditional expression that must evaluate to "true" or "1" for that clock edge to be used as a strobe. The expression is evaluated at the time of the clock edge, rather than after the delay has been applied. Refer to “GUI_expression_format” for legal expression syntax.

  "<expression>" — A conditional expression that results in a true/false value. You must enclose the condition expression within quotation marks (“ ”).

- <clock_name>

  (required) A name for this clock definition. This name will be used with the compare add command when doing a clocked-mode comparison. Must precede the <signal_path> argument.

- <signal_path>

  (required) A full path to the signal whose edges are to be used as the strobe trigger. Must be specified as the final argument to the compare clock command.

Examples

- Define a clocked compare strobe named "strobe" that samples signals on the rising edge of signal gold:.top.clock.

  compare clock -rising strobe gold:.top.clock

Define a clocked compare strobe named "clock12" that samples signals 12 ns after the rising edge of signal gold:/mydesign/clka.

compare clock -rising -delay {12 ns} clock12 gold:/mydesign/clka

Related Topics

- compare add
- Waveform Compare

compare configure

This command modifies options for compare signals and regions. The modified options are applied to all objects in the specified compare path.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

compare configure [-clock <name>] [-recursive] [-tol <time>[<unit>]]
[-tolLead <time>[unit]] [-tolTrail <time>[unit]]
[-vhdlmatches {<ref-logic-value>=<test-logic-value>:…}]
[-vlogmatches {<ref-logic-value>=<test-logic-value>:…}] [-when {“<expression>”}]
<comparePath>

Arguments

- -clock <name>

  (optional) Changes the strobe signal for the comparison. If the comparison is currently asynchronous, it will be changed to clocked. This switch may not be used with the -tol, -tolLead, and -tolTrail options.

  <name>

- -recursive

  (optional) Specifies that signals should also be selected in all nested subregions, and subregions of those, and so forth.

- -tol <time>[<unit>]

  (optional) Specifies the default maximum amount of time the test signal edge is allowed to trail or lead the reference edge in an asynchronous comparison.

  <time> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -tolLead <time>[unit]

  (optional) Specifies the maximum amount of time a test signal edge is allowed to lead a reference edge in an asynchronous comparison.

  <time> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -tolTrail <time>[unit]

  (optional) Specifies the maximum amount of time a test signal edge is allowed to trail a reference edge in an asynchronous comparison.

  <time> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -vhdlmatches {<ref-logic-value>=<test-logic-value>:…}

  (optional) Specifies how VHDL signal states in the reference dataset should match values in the test dataset. Values are specified in a colon-separated list of match values. For example:

  -vhdlmatches {X=XUD:Z=ZD:1=1HD}

  Default is:

  ```
  {U=UWXD:X=UWXD:0=0LD:1=1HD:Z=ZD:W=UWXD:L=0LD:H=1HD:-=UX01ZWLHD}
  ```

- -vlogmatches {<ref-logic-value>=<test-logic-value>:…}

  (optional) Specifies how Verilog signal states in the reference dataset should match values in the test dataset. Values are specified in a colon-separated list of match values. For example:

  -vlogmatches {0=0:1=1:Z=Z}

  Default is:

  ```
  {0=0:1=1:Z=Z:X=X}
  ```

- -when {“<expression>”}

  (optional) Specifies a conditional expression that must evaluate to "true" or "1" for differences to be reported. The expression is evaluated at the start of an observed difference. Refer to “GUI_expression_format” for legal expression syntax.

  "<expression>" — A conditional expression that results in a true/false value. You must enclose the condition expression within quotation marks (“ ”).

- <comparePath>

  (required) Identifies the path of a compare signal, region, or glob expression. Must be specified as the final argument to the compare configure command.

Related Topics

- compare add
- Waveform Compare

compare continue

This command is used to continue with comparison difference computations that were suspended using the compare stop button or Control-C. If the comparison was not suspended, compare continue has no effect.

Syntax

compare continue

Arguments

None

Related Topics

- compare stop
- Waveform Compare

compare delete

This command deletes a comparison object from the currently open comparison.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

compare delete [-recursive] {<objectPath>}

Arguments

- -recursive

  (optional) Deletes a region recursively.

- {<objectPath>}

  (required) Path to the comparison object to be deleted (for example, {*compare:/top/\clk<>clk*}). The comparison object must be "escaped" correctly so the braces '{}' and trailing space are required. Must be specified as the final argument to the compare delete command.

Related Topics

- compare add
- Waveform Compare

compare end

This command closes the active comparison without saving any information.

Syntax

compare end

Arguments

None

Related Topics

- compare add
- Waveform Compare

compare info

This command lists the results of the comparison in the Main window transcript. To save the information to a file, use the -write argument.

Syntax

compare info [-all] [-count] [-primaryonly] [-signals] [-secondaryonly] [<startNum> [<endNum>]] [-summary [-write <filename>]

Arguments

- -all

  (optional) Lists all differences (even those marked as "ignore") in the output. By default, ignored differences are not listed in the output of a compare info command.

- -count

  (optional) Returns the total number of primary differences found.

- -primaryonly

  (optional) Lists only differences on individual bits, ignoring aggregate values such as a bus.

- -signals

  (optional) Returns a Tcl list of compare signal names that have at least one difference.

- -secondaryonly

  (optional) Lists only aggregate value differences such as a bus, ignoring the individual bits.

- <startNum> [<endNum>]

  (optional) Specifies the difference numbers to start and end the list with. If omitted, Questa SIM starts the listing with the first difference and ends it with the last. If just endNum is omitted, Questa SIM ends the listing with the last difference.

- -summary

  (optional) Lists only summary information.

- -write <filename>

  (optional) Saves the summary information to <filename> rather than the Main window transcript. Optional.

Examples

- List all errors in the Main window transcript.

  compare info

- List only an error summary in the Main window transcript.

  compare info -summary

- Write errors 20 through 50 to the file *myerrorfile*.

compare info -write myerrorfile 20 50

Related Topics

- compare add
- compare annotate
- Waveform Compare

compare list

Displays in the Transcript window a list of all the compare add commands currently in effect.

Syntax

compare list [-expand]

Arguments

- -expand

  (optional) Expands groups specified by the compare add command to individual signals.

Related Topics

- compare add
- Waveform Compare

compare options

This command sets defaults for various waveform comparison commands. Those defaults are used when other compare commands are invoked during the current session. To set defaults permanently, edit the appropriate PrefCompare() Tcl variable.

Refer to "Setting GUI Preferences" for details.

If no arguments are used, compare options returns the current setting for all options. If one option is given that requires a value, and if that value is not given, compare options returns the current value of that option.

Syntax

compare options [-addwave] [-all] {[-in] [-inout] [-out] | [-ports]} [-internal] [-noaddwave] [-ignoreVlogStrengths] [-noignoreVlogStrengths] [-maxsignal <n>] [-maxtotal <n>] [-listwin <name>] [-separator <string>] [-tol <time>[<unit>]] [-tolLead <time>[<unit>]] [-tolTrail <time>[<unit>]] [-track] [-notrack] [-vhdlmatches {<ref-logic-value>=<test-logic-value>:…}] [-vlogmatches {<ref-logic-value>=<test-logic-value>:…}] [-wavepane <n>] [-wavewin <name>]

Arguments

- -addwave

  (optional) Specifies that new comparison objects are added automatically to the Wave window. (default) You can specify that objects aren't added automatically using the -noaddwave argument. Related Tcl variable is PrefCompare(defaultAddToWave).

- -all

  (optional) Specifies comparison of all signals.

- -in

  (optional) Specifies comparison of ports of mode IN. Multiple modes can be specified in the same command.

- -inout

  (optional) Specifies comparison of ports of mode INOUT. Multiple modes can be specified in the same command.

- -ignoreVlogStrengths

  (optional) Specifies that Verilog net strengths should be ignored when comparing two Verilog nets. (default) Related Tcl variable is PrefCompare(defaultIgnoreVerilogStrengths).

- -internal

  (optional) Specifies comparison of internal (non-port) signals. Multiple modes can be specified in the same command.

- -listwin <name>

  (optional) Causes specified comparisons to be displayed in the specified List window. Related Tcl variable is PrefCompare(defaultListWindow).

- -maxsignal <n>

  (optional) Specifies an upper limit for the total differences encountered on any one signal. When that limit is reached, Questa SIM stops computing differences on that signal. Related Tcl variable is PrefCompare(defaultMaxSignalErrors).

  <n> — Any positive integer where the default is 100.

- -maxtotal <n>

  (optional) Specifies an upper limit for the total differences encountered. When that limit is reached, Questa SIM stops computing differences. Related Tcl variable is PrefCompare(defaultMaxTotalErrors).

  <n> — Any positive integer where the default is 100.

- -noaddwave

  (optional) Specifies that new comparison objects are not added automatically to the Wave window. The default is to add comparison objects automatically. Related Tcl variable is PrefCompare(defaultAddToWave).

- -noignoreVlogStrengths

  (optional) Specifies that Verilog net strengths should *not* be ignored when comparing two Verilog nets. Related Tcl variable is PrefCompare(defaultIgnoreVerilogStrengths).

- -notrack

  (optional) Specifies that the waveform comparison should *not* track the current simulation. Related Tcl variable is PrefCompare(defaultTrackLiveSim).

- -out

  (optional) Specifies comparison of ports of mode OUT. Multiple modes can be specified in the same command.

- -ports

  (optional) Specifies comparison of all ports. This switch has the same effect as specifying -in, -out, and -inout together.

- -separator <string>

  (optional) Used with the -rebuild option of the compare add command. When a bus has been broken into bits (bit blasted) by a synthesis tool, Questa SIM expects a separator between the base bus name and the bit indication. This option identifies that separator. The default is an underscore (_). For example, the signal "mybus" might be broken down into "mybus_0", "mybus_1", and so forth. Related Tcl variable is PrefCompare(defaultRebuildSeparator).

<string> — Specifies the character(s) to be used as a separator between the base bus name and the bit indication where the default is an underline (_).

- -tol <time>[<unit>]

  (optional) Specifies the default maximum amount of time the test signal edge is allowed to trail or lead the reference edge in an asynchronous comparison.

  You can specify different values for the leading and trailing tolerances using -tolLead and -tolTrail.

  <time> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -tolLead <time>[<unit>]

  (optional) Specifies the default maximum amount of time the test signal edge is allowed to lead the reference edge in an asynchronous comparison. Related Tcl variables are PrefCompare(defaultLeadTolerance) and PrefCompare(defaultLeadUnits).

  <time> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -tolTrail <time>[<unit>]

  (optional) Specifies the default maximum amount of time the test signal edge is allowed to trail the reference edge in an asynchronous comparison. Related Tcl variables are PrefCompare(defaultTrailTolerance) and PrefCompare(defaultTrailUnits).

  <time> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

Graphical representation of tolLead and tolTrail

- -track

  (optional) Specifies that the waveform comparison should track the current simulation. (default) The differences will be updated at the end of each **run** command, so if you want to see differences soon after they occur, use many relatively short run commands. Related Tcl variable is PrefCompare(defaultTrackLiveSim).

- -vhdlmatches {<ref-logic-value>=<test-logic-value>:…}

  (optional) Specifies how VHDL signal states in the reference dataset should match values in the test dataset. Values are specified in a colon-separated list of match values. For example:

  -vhdlmatches {X=XUD:Z=ZD:1=1HD}

  Default is:

  ```
  {U=UWX-:X=UWXD:0=0LD:1=1HD:Z=ZD:W=UWXD:L=0LD:H=1HD:-=UX01ZWLHD}
  ```

  Related Tcl variable is PrefCompare(defaultVHDLMatches).

- -vlogmatches {<ref-logic-value>=<test-logic-value>:…}

  (optional) Specifies how Verilog signal states in the reference dataset should match values in the test dataset. Values are specified in a colon-separated list of match values. For example:

  -vlogmatches {0=0:1=1:Z=Z}

  Default is:

  ```
  {0=0:1=1:Z=Z:X=X}
  ```

  Related Tcl variable is PrefCompare(defaultVLOGMatches).

- -wavepane <n>

  (optional) Specifies the pane of the Wave window in which compare differences will be placed.

  <n> — A positive integer corresponding to the pane of the Wave window. Wave window panes are numbered in sequence starting with 1 for the initial wave window.

- -wavewin <name>

  (optional) Specifies the name of the Wave window in which compare differences will be viewed. Related Tcl variable is PrefCompare(defaultWaveWindow).

  <wname> — The name of the window to which compare differences are to be added.

Examples

- Return the current value of all options.

  compare options

- Set the maxtotal option to 2000 differences.

  compare options -maxtotal 2000

- Return the current value of the maxtotal option.

compare options -maxtotal

- Set the option to ignore Verilog net strengths.

  compare options -ignoreVlogStrengths

- Verilog X will now match X, Z, or 0.

  compare options -vlogxmatches {0=0:1=1:Z=Z:X=XZ0}

- VHDL std_logic X will now match 'U', 'X', 'W', or 'D'.

  compare options -vhdlmatches {X=UXWD}

- Set the leading tolerance for asynchronous comparisons to 300 picoseconds.

  compare options -tolLead {300 ps}

- Set the trailing tolerance for asynchronous comparisons to 250 picoseconds.

  compare options -tolTrail {250 ps}

Related Topics

- compare add
- compare clock
- Waveform Compare

compare reload

This command reloads comparison differences to allow their viewing without recomputation. Prior to invoking compare reload, you must open the relevant datasets with the same names that were used during the original comparison.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

compare reload <rulesFilename> <diffsFilename>

Arguments

- <rulesFilename>

  (required) Specifies the name of the file that was previously saved using the compare saverules command. Must be the first argument to the compare reload command.

- <diffsFilename>

  (required) Specifies the name of the file that was previously saved using the compare savediffs command.

Related Topics

- compare add
- compare run
- compare savediffs
- compare saverules
- compare start
- Waveform Compare

compare reset

This command clears the current compare differences, allowing another compare run command to be executed. Does not modify any of the compare options or any of the signals selected for comparison. This allows you to re-run the comparison with different options or with a modified signal list.

Syntax

compare reset

Arguments

None

Related Topics

- compare add
- compare run
- Waveform Compare

compare run

This command runs the difference computation on the signals selected via a compare add command. Reports in the Transcript window the total number of errors found.

Syntax

compare run [<startTime>[<unit>] [<endTime>[<unit>]]

Arguments

- <startTime>[<unit>]

  (optional) Specifies when to start computing differences. Any positive integer, where the default is zero (0). Simulation resolution can be changed with the -t argument of the vsim command).

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <startTime> and <unit> within curly braces ({}).

- <endTime>[<unit>]

  (optional) Specifies when to end computing differences. Any positive integer, where the default is the end of the dataset simulation run that ends earliest.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <endTime> and <unit> within curly braces ({}).

Examples

- Compute differences over the entire time range.

  compare run

- Compute differences from 5.3 nanoseconds to 57 milliseconds.

  compare run {5.3 ns} {57 ms}

Related Topics

- compare add
- compare end
- compare start
- Waveform Compare

compare savediffs

This command saves the comparison results to a file for later reloading. To be able to reload the file, you must also save the comparison setup using the compare saverules command.

Syntax

compare savediffs <diffsFilename>

Arguments

- <diffsFilename>

  (required) Specifies the name of the file to create.

Related Topics

- compare add
- compare reload
- compare saverules
- Waveform Compare

compare saverules

This command saves the comparison setup information (or "rules") to a file that can be re-executed later. The command saves compare options, clock definitions, and region and signal selections.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

compare saverules [-expand] **<rulesFilename>**

Arguments

- -expand

  (optional) Expands groups specified by the compare add command to individual signals. If you added a region with the compare add command and then deleted signals from that region, you must use the -expand argument or the rules will not reflect the signal deletions.

- <rulesFilename>

  (required) Specifies the name of the file to which you want to save the rules. To load the file at a later time, use the compare reload command. Must be specified as the final argument to the compare saverules command.

Related Topics

- compare add
- compare reload
- compare savediffs
- Waveform Compare

compare see

This command displays the specified comparison difference in the Wave window using whatever horizontal and vertical scrolling are necessary. The signal containing the specified difference will be highlighted, and the active cursor will be positioned at the starting time of the difference.

Syntax

compare see [-first] [-last] [-next] [-nextanno] [-previous] [-prevanno] [-wavepane <n>] [-wavewin <name>]

Arguments

- -first

  (optional) Shows the first difference, ordered by time. Performs the same action as the Find First Difference button in the Wave window.

- -last

  (optional) Shows the last difference, ordered by time. Performs the same action as the Find Last Difference button in the Wave window.

- -next

  (optional) Shows the next difference (in time) after the currently selected difference. Performs the same action as the Find Next Difference button in the Wave window.

- -nextanno

  (optional) Shows the next annotated difference (in time) after the currently selected difference. Performs the same action as the Next Annotated Difference button in the Wave window.

- -previous

  (optional) Shows the previous difference (in time) before the currently selected difference. Performs the same action as the Previous Difference button in the Wave window.

- -prevanno

  (optional) Shows the previous annotated difference (in time) before the currently selected difference. Performs the same action as the Previous Annotated Difference button in the Wave window.

- -wavepane <n>

  (optional) Specifies the pane of the Wave window in which the differences will be placed.

  <n> — A positive integer corresponding to the pane of the Wave window. Wave window panes are numbered in sequence starting with 1 for the initial wave window.

- -wavewin <name>

  (optional) Adds objects to a specified window. Used to specify a particular window when multiple instances of that window type exist.

<name> — The name of the window to which objects are to be added.

Examples

- Show the earliest difference (in time) in the default Wave window.

  compare see -first

- Show the next difference (in time) in the default Wave window.

  compare see -next

Related Topics

- compare add
- compare run
- Waveform Compare

compare start

This command begins a new dataset comparison. The datasets that you'll be comparing must already be open.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

compare start [-batch] [-maxsignal <n>] [-maxtotal <n>] [-refDelay <delay>[<unit>]] [-testDelay <delay>[<unit>]] [**<reference_dataset>**] <test_dataset>

Arguments

- -batch

  (optional) Specifies that comparisons will not be automatically inserted into the Wave window.

- -maxsignal <n>

  (optional) Specifies an upper limit for the total differences encountered on any one signal. When that limit is reached, Questa SIM stops computing differences on that signal. You can change the default using the compare options command or by editing the PrefCompare(defaultMaxSignalErrors) variable in the *pref.tcl* file.

  <n> — Any integer where the default is 100.

- -maxtotal <n>

  (optional) Specifies an upper limit for the total differences encountered. When that limit is reached, Questa SIM stops computing differences. You can change the default using the compare options command or by editing the PrefCompare(defaultMaxTotalErrors) variable in the *pref.tcl* file.

  <n> — Any integer where the default is 100.

- -refDelay <delay>[<unit>]

  (optional) Delays the reference dataset relative to the test dataset. Delays are applied to signals specified with the compare add command. For each signal compared, a delayed virtual signal is created with "_d" appended to the signal name, and these are the signals viewed in the Wave window comparison objects. The delay is not applied to signals specified in compare "when" expressions.

  <delay> — Any non-negative integer.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <delay> and <unit> within curly braces ({}).

- -testDelay <delay>[<unit>]

  (optional) Delays the test dataset relative to the reference dataset. Delays are applied to signals specified with the compare add command. For each signal compared, a delayed virtual signal is created with "_d" appended to the signal name, and these are the signals viewed in the Wave window comparison objects. The delay is not applied to signals specified in compare "when" expressions.

  <delay> — Any non-negative integer.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <delay> and <unit> within curly braces ({}).

- <reference_dataset>

  (required) The dataset to be used as the comparison reference. Must be specified as the final argument to the compare start command unless you also specify the <test_dataset> argument.

- <test_dataset>

  (optional) The dataset to be tested against the reference. If not specified, Questa SIM uses the current simulation. The reference and test datasets may be the same.

Examples

- Begin a waveform comparison between a dataset named "gold" and the current simulation. Assumes the gold dataset was already opened.

  ```
  compare start gold
  ```

- This command sequence opens two datasets and starts a comparison between the two using greater than default limits for total differences encountered.

  ```
  dataset open gold_typ.wlf gold
  dataset open bad_typ.wlf test
  compare start -maxtotal 5000 -maxsignal 1000 gold test
  ```

Related Topics

- compare add
- compare options
- compare stop
- Waveform Compare

compare stop

This command is used internally by the compare stop button to suspend comparison computations in progress. If a compare run execution has returned to the VSIM prompt, compare stop has no effect. Under Unix, entering a Control-C character in the window that invoked Questa SIM has the same effect as compare stop.

Syntax

compare stop

Arguments

None

Related Topics

- compare run
- compare start
- Waveform Compare

compare update

This command is primarily used internally to update the comparison differences when comparing a live simulation against a *.wlf* file. The compare update command is called automatically at the completion of each simulation run if the compare options -track is in effect.

The user can also call compare update periodically during a long simulation run to cause difference computations to catch up with the simulation. This command does nothing if the compare option -track command was not in effect when the compare run command was executed.

Syntax

compare update

Arguments

None

Related Topics

- compare run
- compare options
- Waveform Compare

configure

The configure command invokes the List or Wave widget configure command for the current default List or Wave window.

To change the default window, use the view command.

Some arguments to this command are order-dependent. Please read through the arguments for further information.

Syntax

configure **list | wave** [-window <wname>] [<option> <value>]

---- List Window Arguments
[-delta [all | collapse | events | none]] [-gateduration [<duration_open>]]
[-gateexpr [<expression>]] [-usegating [off | on]] [-strobeperiod [<period>[<unit>]]]
[-strobestart [<start_time>[<unit>]]] [-usesignaltriggers [0 | 1]] [-usestrobe [0 | 1]]

---- Wave Window Arguments
[-childrowmargin [<pixels>]] [-cursorlockcolor [<color>]] [-gridauto [off | on]]
[-gridcolor [<color>]][-griddelta [<pixels>]] [-gridoffset [<time>[<unit>]]]
[-gridperiod [<time>[<unit>]]] [-namecolwidth [<width>]] [-rowmargin [<pixels>]]
[-signalnamewidth [<value>]] [-timecolor [<color>]] [-timeline [0 | 1]]
[-timelineunits [fs | ps | ns | us | ms | sec | min | hr]] [-valuecolwidth [<width>]]
[-vectorcolor [<color>]] [-waveselectcolor [<color>]] [-waveselectenable [0 | 1]]

Description

The command works in three modes:

- without options or values it returns a list of all attributes and their current values
- with just an option argument (without a value) it returns the current value of that attribute
- with one or more option-value pairs it changes the values of the specified attributes to the new values

The returned information has five fields for each attribute: the command-line switch, the Tk widget resource name, the Tk class name, the default value, and the current value.

Arguments

- list | wave

  (required) Controls the widget to configure. Must be specified as the first argument to the configure command.

  list — Specifies the List widget.

  wave — Specifies the Wave widget.

- -window <wname>

  (optional) Specifies the name of the List or Wave window to target for the configure command. (The view command allows you to create more than one List or Wave window). If no window is specified the default window is used; the default window is determined by the most recent invocation of the view command and has “ - Default” appended to the name.

- <option> <value>

  -bg <color> — (optional) Specifies the window background color.

  -fg <color> — (optional) Specifies the window foreground color.

  -selectbackground <color> — (optional) Specifies the window background color when selected.

  -selectforeground <color> — (optional) Specifies the window foreground color when selected.

  -font <font> — (optional) Specifies the font used in the widget.

  -height <pixels> — (optional) Specifies the height in pixels of each row. .

Arguments, List window only

- -delta [all | collapse | events | none]

  (optional) Specifies how information is displayed in the delta column. To use -delta, -usesignaltriggers must be set to 1 (on).

  all — Displays a new line for each time step on which objects change.

  collapse — Displays the final value for each time step.

  events — Displays an "event" column rather than a "delta" column and sorts List window data by event.

  none — Turns off the display of the delta column.

- -gateduration [<duration_open>]

  (optional) Extends gating beyond the back edge (the last list row in which the expression evaluates to true). The duration for gating to remain open beyond when -gateexpr (below) becomes false, expressed in x number of timescale units. The default value for normal synchronous gating is zero. If -gateduration is set to a non-zero value, a simulation value will be displayed after the gate expression becomes false (if you don’t want the values displayed, set -gateduration to zero).

  <duration_open> — Any non-negative integer where the default is 0 (values are not displayed).

- -gateexpr [<expression>]

  (optional) Specifies the expression for trigger gating. (Use the -usegating argument to enable trigger gating.) The expression is evaluated when the List window would normally have displayed a row of data.

  <expression> — An expression. Refer to the GUI_expression_format for information on expression syntax.

- -usegating [off | on]

  (optional) Enables triggers to be gated on or off by an overriding expression. (Use the -gatexpr argument to specify the expression.) Refer to "Using Gating Expressions to Control Triggering" for additional information on using gating with triggers.

  off — (default) Triggers are gated off (a value of 0).

  on — Triggers are gated on (a value of 1).

- -strobeperiod [<period>[<unit>]]

  (optional) Specifies the period of the list strobe.

  <period> — Any non-negative integer.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <delay> and <unit> within curly braces ({}).

- -strobestart [<start_time>[<unit>]]

  (optional) Specifies the start time of the list strobe.

  <start_time> — Any non-negative integer.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <delay> and <unit> within curly braces ({}).

- -usesignaltriggers [0 | 1]

  (optional) Specifies whether or not signals are to be used as triggers.

  0 — Signals are not used as triggers

  1 — Signals are used as triggers

- -usestrobe [0 | 1]

  (optional) Specifies whether or not a strobe is used as a trigger.

  0 — Strobe is not used to trigger.

  1 — Strobe is used to trigger.

Arguments, Wave window only

- -childrowmargin [<pixels>]

  (optional) Specifies the distance in pixels between child signals. Related Tcl variable is PrefWave(childRowMargin).

  <pixels> — Any non-negative integer where the default is 2.

- -cursorlockcolor [<color>]

  (optional) Specifies the color of a locked cursor. Related Tcl variable is PrefWave(cursorLockColor).

<color> — Any Tcl color where the default is red.

- -gridauto [off | on]

  (optional) Controls the grid period when in simulation time mode.

  off — (default) user-specified grid period is used.

  on — grid period is determined by the major tick marks in the time line.

- -gridcolor [<color>]

  (optional) Specifies the background grid color. Related Tcl variable is PrefWave(gridColor).

  <color> — Any color where the default is grey50.

- -griddelta [<pixels>]

  (optional) Specifies the closest (in pixels) two grid lines can be drawn before intermediate lines will be removed. Related Tcl variable is PrefWave(gridDelta).

  <pixels> — Any non-negative integer where the default is 40.

- -gridoffset [<time>[<unit>]]

  (optional) Specifies the time (in user time units) of the first grid line. Related Tcl variable is PrefWave(gridOffset).

  <time> — Any non-negative integer where the default is 0.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <delay> and <unit> within curly braces ({}).

- -gridperiod [<time>[<unit>]]

  (optional) Specifies the time (in user time units) between subsequent grid lines. Related Tcl variable is PrefWave(gridPeriod).

  <time> — Any non-negative integer where the default is 1.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <delay> and <unit> within curly braces ({}).

- -namecolwidth [<width>]

  (optional) Specifies the width of the name column in pixels. Related Tcl variable is PrefWave(nameColWidth).

  <width> — Any non-negative integer where the default is 150.

- -rowmargin [<pixels>]

  (optional) Specifies the distance between top-level signals in pixels. Related Tcl variable is PrefWave(rowMargin).

<pixels> — Any non-negative integer where the default is 4.

- -signalnamewidth [<value>]

  (optional) Controls the number of hierarchical regions displayed as part of a signal name shown in the pathname pane. Related Tcl variable is PrefWave(SignalNameWidth). Can also be set with the WaveSignalNameWidth variable in the *modelsim.ini* file.

  <value> — Any non-negative integer where the default is 0 (display the full path. For example,1 displays only the leaf path element, 2 displays the last two path elements, and so on.

- -timecolor [<color>]

  (optional) Specifies the time axis color. Related Tcl variable is PrefWave(timeColor).

  <color> — Any color where the default is green.

- -timeline [0 | 1]

  (optional) Specifies whether the horizontal axis displays simulation time or grid period count. Related Tcl variable is PrefWave(timeline).

  0 — (default) Simulation time is displayed.

  1 — Grid period count is displayed.

- -timelineunits [fs | ps | ns | us | ms | sec | min | hr]

  (optional) Specifies units for timeline display. Does not affect the currently-defined simulation time.

  fs — femtosecond (10^{-15} seconds)

  ps — picosecond (10^{-12} seconds)

  ns — nanosecond (10^{-9} seconds) (default)

  us — microsecond (10^{-6} seconds)

  ms — millisecond (10^{-3} seconds)

  sec — second

  min — minute (60 seconds)

  hr — hour (3600 seconds)

- -valuecolwidth [<width>]

  (optional) Specifies the width of the value column, in pixels. Related Tcl variable is PrefWave(valueColWidth).

  <width> — Any non-negative integer where the default is 100.

- -vectorcolor [<color>]

  (optional) Specifies the vector waveform color. Default is #b3ffb3. Related Tcl variable is PrefWave(vectorColor).

  <color> — Any color where the default is #b3ffb3.

- -waveselectcolor [<color>]

  (optional) Specifies the background highlight color of a selected waveform. Related Tcl variable is PrefWave(waveSelectColor).

  <color> — Any color where the default is grey30.

- -waveselectenable [0 | 1]

  (optional) Specifies whether the waveform background highlights when an object is selected. Related Tcl variable is PrefWave(waveSelectEnabled).

  0 — (default) Highlighting is disabled.

  1 — Highlighting is enabled.

To get a more readable listing of all attributes and current values, use the lecho command, which pretty-prints a Tcl list.

There are more options than are listed here. See the output of a configure list or configure wave command for all options.

Examples

- Display the current value of the strobeperiod attribute.

  config list -strobeperiod

- Set the period of the list strobe and turns it on.

  config list -strobeperiod {50 ns} -strobestart 0 -usestrobe 1

- Set the wave vector color to blue.

  config wave -vectorcolor blue

- Set the display in the current Wave window to show only the leaf path of each signal.

  config wave -signalnamewidth 1

Related Topics

- view
- Setting GUI Preferences

context

This command provides several operations on a context's name. The option you specify determines the operation.

For some of the commands there is a required argument "<name>". Please read through the arguments for further information.

Syntax

context children <name>

context current <name>

context dataset

context du <name>

context exists <name>

context fullpath <name>

context isCell <name>

context isHierBlock <name>

context isInst <name>

context isLoggable <name>

context isNet <name>

context isPrimitive <name>

context isProc <name>

context isProcessScope <name>

context isScope <name>

context isVar <name>

context isVhdl <name>

context join <name> <name> ...

context name <name>

context parent <name>

context path <name>

context split <name>

context tail <name>

context type <name>

context visibility <name>

Arguments

- children <name>

  Returns a list of declared generics, parameters, signals, nets, and scopes within the named instance.

  <name> — (required) Name of a valid instance.

- current <name>

  Returns the currently active instance [env] for the given dataset.

  <name> — (required) Name of the active dataset or dataset alias.

- dataset

  Returns the open dataset name.

- du <name>

  Returns the following design unit information: library path, primary design unit name, and secondary design unit name for the named instance.

  <name> — (required) Name of a valid design unit.

- exists <name>

  Returns true (1) if <name> exists in the design.

  <name> — (required) Name of a context object or region. Does not have to be a valid object name.

- fullpath <name>

  Returns the full path (including the dataset prefix) of the specified <name>.

  <name> — (required) Name of a context object or region.

- isCell <name>

  Returns true (1) if the specified instance is a cell.

  <name> — (required) Name of a ‘celldefine or vital cell.

- isHierBlock <name>

  Returns true (1) if the named instance is a scope.

  <name> — (required) Name of a block scope or generate scope.

- isInst <name>

  Returns true (1) if <name> is an instance pathname.

  <name> — (required) Name of a context object or region. Does not have to be a valid object name.

- isLoggable <name>

  Returns true (1) if the named object is loggable with the log, add wave, and add list commands.

<name> — (required) Name of a valid object.

- isNet <name>

  Returns true (1) if <name> is a Signal or Net pathname.

  <name> — (required) Name of a context object or region. Does not have to be a valid object name.

- isPrimitive <name>

  Returns true (1) if the named instance is a verilog primitive.

  <name> — (required) Name of a valid instance.

- isProc <name>

  Returns true (1) if <name> is valid Process pathname.

  <name> — (required) Name of a context object or region. Does not have to be a valid object name.

- isProcessScope <name>

  Returns true (1) if the specified name is a process scope.

  <name> — (required) Name of a valid instance.

- isScope <name>

  Returns true (1) if the specified name is a scope or instance name.

  <name> — (required) Name of an instance or scope.

- isVhdl <name>

  Returns true (1) if the named object is a VHDL object.

  <name> — (required) Name of an object.

- join <name> <name> ...

  Takes one or more <name> and combines them, using the correct path separator.

  <name> — (required) Name of a context object or region. Does not have to be a valid object name.

- parent <name>

  Returns the parent path of <name> by removing the tail (see context tail).

  <name> — (required) Name of a context object or region.

- path <name>

  Returns the pathname portion of <name>, removing the dataset name.

  <name> — (required) Name of a context object or region.

- split <name>

  Returns a list whose elements are the path components in <name>. The first element of the list will be the dataset name if one is present in the name, including the dataset separator. For example, context split /foo/bar/baz returns / foo bar baz.

  <name> — (required) Name of a context object or region.

- tail <name>

  Returns all of the characters in <name> after the last path separator. If <name> contains no separators then returns <name>. Any trailing path separator is discarded.

  <name> — (required) Name of a context object or region.

- type <name>

  Returns a string giving the acc type of <name>.

  <name> — (required) Name of a context object or region.

- visibility <name>

  Returns the “acc” visibility settings for the named instance.

  <name> — (required) Name of a context object or region.

coverage analyze

This command is used to display test information based on a merged UCDB, created by a vcover merge operation. Some arguments for this command require that the merge to create the data was performed with the test-associated merge, by running vcover merge -testassociated. By default, merges are performed without this information.

This command is only available during post-simulation processing, when a UCDB file is opened with vsim -viewcov.

For all -select arguments: the selection criteria of -eq (equals) includes matched items in the results; -ne (not equal) excludes the matched items.

The coverage statistics displayed in the output of this command are calculated in accordance with the coverage aggregation algorithm shown in "Calculation of Total Coverage".

Alias nodes are signified by a '+' sign that is appended to the node name.

Syntax

coverage analyze <miscellaneous coverage and filter args> <output mode args>

where <miscellaneous coverage and filter args> are:

Miscellaneous Arguments

[-precision <int>] [-testextract <testspec>+] [-file <filename> [-append]]

Arguments to specify results by coverage type

-code {b | c | e | f | s | t}...[-codeAll] [-assert] [-cvg] [-directive] [-select instance]

Arguments to specify coverage by path or context

[[-du <spec>] [-path <obj>]] | [-plansection <obj> [-section] | [-nosection] |
[-showattribute <attribute_key_string>]] [-recursive]

Arguments to specify by query of results

[-select attr <name> {([-lt|-gt|-le|-ge]<int>)|-eq|-ne|-[n]regexp} {<value>}]
[-select coverage {-lt|-gt|-le|-ge|-eq|-ne} <float>]
[-select dateattribute <name> {-lt|-gt|-le|-ge|-eq|-ne} {<value>}]
[-select instance]
[-select name {-eq|-ne|-[n]regexp} < string >]
[-select tag {-eq|-ne|-[n]regexp} <string>]
[-select unlinked]
[-select weight {-lt|-gt|-le|-ge|-eq|-ne} <int>]
[-anyselected | -allselected] [-aggregate preselected | postselected]
[-hideexcluded] [-nozeroweights] [-prunechildren | -noprunechildren]

and where <output mode args> are:

Specifying output mode (mutually exclusive)

[-coverage most | least | zero | nonzero]
| [-samples most | least | zero | nonzero]

| [-summary [hier | local]]
| [-total]

Arguments

- -aggregate preselected | postselected

  (optional: Only has an effect with the -select options (other than instance), can only be used with -plansection.)

  Compute coverage for parent nodes before the selection of children, in other words based on *all* children (preselected)-- or after the selection, in other words based only on *selected* children (postselected).

- -anyselected | -allselected

  (optional: Only has an effect with multiple -select options (does not work with -select instance)). These switches are mutually exclusive and only one may be given on each command line.) Use any select criteria (logical OR) or all select criteria (logical AND) to determine what is displayed in the report. Default is -allselected.

- -append

  (optional) Appends to output file, if -file option was given.

- -assert

  (optional) Reports tests in which assertions contributed to coverage data. Assertions contribute to coverage based on "% non-vacuous passes".

- -code {b | c | e | f | s | t}...

  (optional) Reports tests in which a particular code coverage type (branch, condition, expression, statement, toggle, FSM) contributed to coverage.

- -codeAll

  (optional) Reports tests in which any coverage type contributed to coverage. Equivalent to -code bcestf.

- [-coverage most | least | zero | nonzero]

  (optional) Reports which test yields most / least / zero / coverage for the specified parameters. This argument is on a per test basis. This argument requires that a test-associated merge (vcover merge -testassociated) has been previously executed.

- -cvg

  (optional) Specifies the command apply to covergroup data.

- -directive

  (optional) Specifies the command apply to directive data.

- -du <spec>

  (optional, though -plansection, -du or -path must be specified.)

  Restricts analyze results to specified design units.

- -file <filename>

  (optional) Write to output file.

- -hideexcluded

  (optional) Prevents any excluded coverage items from appearing in the report.

- -nosection

  (optional) Excludes section numbers (specified with -plansection) from the test information displayed. Allows for more concise output. Mutually exclusive with -section.

- -nozeroweights

  (optional) Prevents any items with zero weights from appearing in the report, including all zero-weighted coverage items and testplan sections.

  Has no effect on coverage numbers.

- -path <obj>

  (optional, though -plansection, -du or -path must be specified.)

  Restricts coverage analysis results to design (non-testplan) paths matching the specified object. The <obj> can be used to specify a dataset other than the current dataset. (Refer to "Object Name Syntax" for instructions on how to specify a dataset.) If no dataset is specified, the current dataset is used. Only one dataset name per command invocation may be used or an error will result. This switch can be combined with the -du switch to be du-relative.

- -plansection <obj>

  (optional) Restricts the analyze results to the given testplan path. By default, the query applies to all testplan scopes in the database. It also applies to all linked coverage, instance, or design unit scopes. Either -du, -path, or -plansection must be specified.

- -precision <int>

  (optional) Precision for output.

- -prunechildren | -noprunechildren

  (optional) The -prunechildren switch prunes children from nodes surviving a selection (after -select is applied) from the report. By default (-noprunechildren), the children of nodes are included in the coverage analyze report: the report includes all children of coverage items or testplan sections that survive a -select filter. This is useful in cases where you want to subdivide a testplan report by category or area of responsibility.

- -recursive

  (optional) Specifies that the analyze is applied recursively. The default is for the query to be restricted to the single object or objects specified in the command.

- [-samples most | least | zero | nonzero]

  (optional) Reports which test has the optionally computed sample count, meeting the specified criteria. The specified criteria are: most samples, least samples, no samples, or any

samples. This argument is on a per test basis. Using this argument implies the use of -cvg, since only covergroups and covergroup instances have sample counts. This argument is only compatible with a merge (vcover merge) performed with -testassociated.

- -section

  (optional) Includes section numbers (specified with -plansection) in the test information displayed. This is the default. Mutually exclusive with -nosection.

- -select attr <name> {([-lt|-gt|-le|-ge]<int>)|-eq|-ne|-[n]regexp} {<value>}

  (optional) Displays user attributes in the report. The <name> of the user-defined attribute is a string and must be given as the second argument. The <value> can be a float, integer or string. The following rules apply to name/value pairs:

  - -lt|-gt|-le|-ge require <int>, an integer numerical argument.

    -lt is "less than", -gt is "greater than", -ge is "greater than or equal to", -le is "less than or equal to", -eq is "equal to", and -ne is "not equal to", -regexp is "regular expression match", -nregexp is "regular expression does not match".

  - -regexp and -nregexp require a string argument.

  - The -regexp option uses Tcl regular expression matching (as in the memory window); this offers alternative and more sophisticated matching. One crucial difference is that "match anything starting with 'a'" is expressed differently in the two semantics: as "-eq a*" and "-regexp ^a.*".

  - For -eq or -ne, the argument is held as a string and will match int or float or double attribute values as appropriate (for example, "2" could match any type attribute).

  - In cases where an attribute is of a type incompatible with the specified value, a single warning is issued.

- -select dateattribute <name> {-lt|-gt|-le|-ge|-eq|-ne} {<value>}

  (optional) Selects results whose dates match the criteria specified. The <name> must be a string and must be given as the second argument. The <value> must also a string (the date being selected for filtering). The following rules apply to name/value pairs:

  - -lt|-gt|-le|-ge|-eq|-ne

    -lt is "less than", -gt is "greater than", -ge is "greater than or equal to", -le is "less than or equal to", -eq is "equal to", and -ne is "not equal to".

  - <value> is the date you are selecting (or filtering out) and the formats accepted are:

  a. mm/dd[/yy]

  b. monthname dd[, yy]

  c. dd monthname [yy]

  d. [yy]yymmdd (for example, trend date)

e. [yy]yy-mm-dd

f. dd-monthname-[yy]yy

In the formats where the year specification is optional (formats a, b, c above), the default is the current year. Abbreviations are allowed for months, and upper and/or lowercase is accepted.

- -select coverage {-lt|-gt|-le|-ge|-eq|-ne} <float>

  (optional) Includes the coverage of the items in the test information displayed.

- -select instance

  (optional) Includes design instances only in the report: Verilog module, program, and interface instances, VHDL architectures, and package instances.

- -select name {-eq|-ne|-[n]regexp} < string >

  (optional) Displays all unlinked coverage and/or testplan items, as specified with either the -plan, -path or -du argument. The -select name filter must be used with -eq, -ne, or -[n]regexp arguments and a string, and <string> can be a pattern with wildcards.

  - -eq is "equal to"
  - -ne is "not equal to"
  - -regexp is "regular expression match", and -nregexp is "regular expression does not match"

    The -regexp and -nregexp options use Tcl regular expression matching.

- -select unlinked

  (optional) Reports on objects that are "unlinked" for the purposes of test traceability. See coverage unlinked command for further details.

- -select tag {-eq|-ne|-[n]regexp} <string>

  (optional) Reports the tag used to associate the specified object with the testplan, or to associate the object with the other objects. The tag is assigned with the coverage tag command. Tags can be used for primitive (non-hierarchical) grouping of coverage in the database. Tags are used for testplan to coverage linking as well.

  - tag can only be used with -eq, -ne, or -[n]regexp and a string. <string> can be a pattern with wildcards.

    -eq is "equal to", and -ne is "not equal to", -regexp is "regular expression match", -nregexp is "regular expression does not match".

  - The -regexp option uses Tcl regular expression matching (as in the memory window); this offers alternative and more sophisticated matching. One crucial difference is that "match anything starting with 'a'" is expressed differently in the two semantics: as "-eq a*" and "-regexp ^a.*".

- -select weight {-lt|-gt|-le|-ge|-eq|-ne} <int>

  (optional) Includes the weights of the coverage objects in the test information displayed. You set the weight for coverage objects using the coverage weight (-du|-path|-plan) command.

  - -lt is "less than", -gt is "greater than", -ge is "greater than or equal to", -le is "less than or equal to", -eq is "equal to", and -ne is "not equal to", -regexp is "regular expression match", -nregexp is "regular expression does not match". Optional.

- -showattribute <attribute_key_string>

  (optional) Reports the specified testplan attribute. Can only be used with -plansection. Multiple instances are allowed. Only testplan attribute keys may be used. All other values of "attribute_key_string" will result in a warning message. Wildcards are not supported.

- -summary [hier | local]

  (optional) Generates a summary (stats) style report for the given path(s). The default argument to -summary is "hier".

  - "hier" generates statistics recursively including sub-instances of the design hierarchy, if used with -path. If used with -plansection, "hier" is testplan section recursive.
  - "local" generates statistics for the local instance only.

  This argument can not be used with -du. If used with -path, the -select instance argument is applied automatically, which restricts the command to design instance scopes; the summary report is most useful (for example, shows multiple kinds of coverage) if used with design instances. For other types of coverage, for example, covergroup and FSM coverage hierarchy, the -totals report is much more concise.

  The summary report generated by this option includes Assertion Passes and/or Successes, Failures, and Attempts, depending on whether -assertcover is present. See vsim -assertcover for detials.

- -testextract <testspec>+

  (optional) For a vcover merge, this reconstructs a report for the specified test or tests. <testspec> is a list of TESTNAME attribute names for all tests in the UCDB. Merge must have been performed with the -testassociated switch to vcover merge.

- -total

  (optional) Generates a hierarchical report of coverage totals — as a single number — for testplan sections, based on children nodes and linked coverage items and instances or design units (those sharing a tag with the testplan scope). When used non-recursively, -total supports bin paths. Could also be used for HDL design unit or instance scopes. It calculates the average of all different types of coverage found, for example, an average of results obtained with
  -summary. For instances or design units, the total coverage is an average of all types. The "coverage" is the total coverage number computed for a coverage object:

- Weighted average of different kinds of coverage found within a design instance or design unit.
  - Design units only "contain" code coverage, not functional coverage or assertions.
  - The global per-type weights set with coverage weight can be used to affect the weighted average, but these weights apply globally.
- Coverage result is for a particular kind of coverage object: FSM, covergroups, cover directive, toggle, branch, and so forth.
- Non-vacuous pass result (100% if any, 0% if none) for an assertion.

Hierarchically computed weighted average of children of a testplan section, can be computed pre or post selection with -aggregate preselected|postselected.

Examples

- Display all coverage items in a specified path:

```
coverage analyze -path /top/*
```

You can such a command to find the path for any particular coverage object in /top, such as a statement.

- List tests which covered a particular statement:

```
coverage analyze -path /top/#stmt#11#1# -coverage nonzero
```

Returns:

```
#
# Tests with Non-Zero Coverage:
#
# /top/#stmt#11#1#
#     test1_0                  100.00%
#
```

- List tests which hit a specific coverage point:

```
coverage analyze -path /top/dut/u1/cvg_inst/cp1 -coverage nonzero
```

- List tests which hit a specific coverage bin:

```
coverage analyze -path /top/dut/u1/cvg_inst/cp1/bin1 -coverage nonzero
```

- List tests which did not hit a specific coverage bin:

```
coverage analyze -path /top/dut/u1/cvg_inst/cp3/bin2 -coverage zero
```

- Generate a hierarchical report of all design units:

```
coverage analyze -du *-totals -r
```

- Generate a hierarchical report of all testplan scopes, including linked scopes:

```
vsim> coverage analyze -plansection / -totals -r

Total Coverage Report
Sec#     Testplan Section / Coverage Link       Coverage    Goal Weight
----------------------------------------------------------------------
-        /testplan                                56.18% 100.00%      1
1          Top                                    56.18% 100.00%      1
1.1          Bits                                 85.24% 100.00%      1
               /top/child1/cvg_bits_vs_clock      83.33% 100.00%      2
               /top/child1/cvg_bits_vs_bits       89.06% 100.00%      1
1.2          Arithmetic                           27.11% 100.00%      1
               /top/child2/cvg_arith              27.11% 100.00%      1
```

- Various examples of testplan queries:

  Is there a test which has zero coverage for the named testplan?

  ```
  vsim> coverage analyze -plansection /test1.0/test1.1 -coverage zero
    test0
    test4
  ```

  Show the test which has least coverage for the named testplan. Show summary.

  ```
  vsim> coverage analyze -plansection /test1.0/test1.2 -coverage least
                          -summary
    test13:
    Coverage Summary BY INSTANCES ...
  ```

  Show the test which has most coverage for the named testplan. Show total only.

  ```
  vsim> coverage analyze -plansection /test1.0/test1.3 /
  -coverage most -total test27: 99%
  ```

  Filter by cover directives only:

  ```
  vsim> coverage analyze -plansection /test1.0/test1.3 /
  -coverage most -total -dir test27: 90%
  ```

  Ask the question based on a particular covergroup:

  ```
  vsim> coverage analyze -path /top/i/cvg1 -coverage most /
  -total test23: 85%
  ```

  Display anything not completely covered in a testplan:

  ```
  coverage analyze -plansection / -r -select cover -lt 100
  ```

  Display coverage holes with weight greater than 1:

  ```
  coverage analyze -plansection / -r -select cover -eq 0 -weight -gt 1
  ```

  Display a list of items which are less than 100% covered within the top level plan section, belonging to a particular engineer:

  ```
  coverage analyze -plansection / -r -select cover -lt 100 /
  -select Engineer -eq mike
  ```

  Match a user-defined attribute "myattr" whose value starts with "a", or one which does not start with "a":

```
coverage analyze -plansection / -r -select attr myattr /
-regexp "^a.*"

coverage analyze -plansection / -r -select attr myattr /
-nre "^a.*"
```

- Various examples of using the -select name option:

  coverage analyze -r -select name -eq {/top/mach/state_out}

  coverage analyze -r -select name -eq {/top/mach/state_out} -prunechildren

  coverage analyze -r -select name -eq /top/*/state — [Here, * means "match any number of any characters"]

  coverage analyze -r -select name -ne {/top/fib_val}

  coverage analyze -r -select name -regexp prop

  coverage analyze -r -select name -regexp branch

  coverage analyze -r -select name -nregexp mach

  coverage analyze -r -select name -nregexp mach -prunechildren

  coverage analyze -r -select name -regexp /top/tog*.e/sbts — [Here, * means "match any number of the preceding characters (i.e 'g' or 'ggggg')" and . means "match any single character"]

Related Topics

- Code Coverage
- Calculation of Total Coverage
- Coverage and Verification Management in the UCDB
- Verification Management Browser Window
- Verification Tracker Window
- Viewing Test Data in the Tracker Window
- Verification Management Overview
- coverage analyze
- coverage attribute
- coverage goal
- coverage report
- coverage save
- coverage tag
- coverage unlinked
- coverage weight
- vcover merge
- vcover ranktest
- vcover report
- vcover stats
- vsim

coverage attribute

The coverage attribute command is used to display or set attributes in the currently loaded database on the following types of attributes:

- Test Attributes — attributes for each test attribute record (one record is created for each simulation that is saved). These attributes are name value pairs that represent testcase information. Refer to the section "Predefined Attribute Data" for complete list of these attributes.
- UCDB Attributes — attached globally to the UCDB file, read or written with "coverage attribute -ucdb". Unlike test attributes, these are merged together during a vcover merge. In the current system, the only attributes created by Questa SIM are those related to the test-associated merge (vcover merge -testassociated). However, you can create attributes for your own use, accessible through this CLI or the UCDB API.
- Object Attributes — attached to particular objects stored in the UCDB (ex. design units, design instance scopes, a particular covergroup, or a particular cover directive). Some attributes for different kinds of objects are created by Questa SIM, but you can create or read any attribute in the CLI or the UCDB API.

This command can be used both during simulation and with "vsim -viewcov", though in simulation it can only be used for test attributes (the single test attribute record that exists in simulation).

To apply filters (-select instance, -assert, -code, and so forth.):

1. Match paths first, with recursion (if specified).
2. Specify paths to be "thrown out" (those not matching the filter).

This command can also be used to create global user attributes and enable them to capture trend data using the -trendable argument.

Syntax

To display, set, or delete test attributes

```
coverage attribute [-test <testname>]
    ([-seed [<num>]] [-command [<str>]] [-comment [<str>]] [-compulsory [0|1]] ) |
    ([-name <str> -value <int> [-name <str> -value <int>]...] [-delete] )
    [-tcl] [-concise]
```

To display, set, or delete UCDB (global) attributes

```
coverage attribute -ucdb [-name <str> -value <int> [-name <str> -value <int>]...] [-delete] [-tcl]
    [-concise]
```

To display, set, or delete object attributes

```
coverage attribute ([<coverage_types>] [-match <str> | -path <obj> | -plansection <obj>]
    [-du <duname>] [-history <history_name>] [-select instance]
    [-name <str> -value <int> [-name <str> -value <int>]...] [-delete] [-tcl] [-concise]
```

Where <coverage_types> =

[-assert] [-code {b | c | e | f | s | t}...] [-codeAll] [-cvg] [-directive]

To create attributes to capture trending data

coverage attribute [-ucdb [-trendable]] -name <str> -value <int>

Arguments

- -assert

  (optional) Specifies this command applies to assertion data.

- -code {b | c | e | f | s | t}...

  (optional) Specifies this command applies to corresponding code coverage types: branch, condition, expression, statement, toggle, FSM.

- -codeAll

  (optional) Specifies this command applies to all coverage types. Equivalent to -code bcestf.

- -command [<str>]

  (optional) Command to run the test: script command line, “knob settings”, and so forth. If <str> is specified, it sets the command to run: If not set, commands used are displayed.

- -comment [<str>]

  (optional) Comment on the testcase. If <str> is specified, sets a comment: If not, comments are displayed.

- -compulsory [0|1]

  (optional) Indicates test is compulsory. By default, it is not compulsory (0).

- -concise

  (optional) Print attribute values only, do not print other information.

- -cvg

  (optional) Specifies the command apply to covergroup data.

- -delete

  (optional) Delete specified name attributes.

- -directive

  (optional) Specifies the command apply to directive data.

- -du <duname>

  (optional) Apply to a design unit, for example, “lib.primary(secondary)” secondary for VHDL only.

- -history <history_name>

  (optional) Displays all attributes on the <history_node_name>. Optionally, can be used with the -name arguments to specify particular attribute or attribute/value pairs to be displayed. With -name and -value, this argument can also be used to overwrite the value of an existing attribute on a specific history node. See examples below.

- -match <str>

  (optional) Recursively matches the given pattern against the specified coverage types in the entire instance tree. If -duname is specified, it matches against the specified coverage types in the design unit. Wildcards are allowed.

- -name <str> -value <int>

  Required for setting attributes; optional for displaying them. Sets attribute name and value. Used to add user defined attributes to either a test, a UCDB, or a particular object in the UCDB. You can specify multiple -name <name> -value <value> pairs per coverage attribute command.

  For more information on these attributes, and a list of predefined attribute fields that exist in a UCDB Test Attribute Record, refer to "Predefined Attribute Data". For example, before you save the test data to a UCDB, you can assign a unique test name for each test run using the command "coverage attr -name TESTNAME -value run_1". Then, when you merge the tests for later analysis, the test data records will remain unique and data conflicts are avoided.

- -path <obj>

  (optional) Apply to a path in the UCDB. The <obj> can be used to specify a dataset other than the current dataset. (Refer to "Object Name Syntax" for instructions on how to specify a dataset.) If no dataset is specified, the current dataset is used. Only one dataset name per command invocation may be used or an error will result. Wildcards are acceptable. Relative path can be used in conjunction with the -du switch.

- -plansection <obj>

  (optional) Apply to a testplan section in the UCDB, as specified by <obj>. Wildcards are acceptable. Relative path can be used in conjunction with -du.

- -seed [<num>]

  (optional) Random seed of the test run. Without the <num>, the seed is displayed. Specifying <num> sets the seed.

- -select instance

  (optional) Restricts the command to apply to design instance scopes (VHDL architectures, interface instances, and so forth.).

- -tcl

  (optional) Prints attribute information in a Tcl format.

- -test <testname>

  (Required when used with vsim -viewcov; optional otherwise.) Specifies a test object for attributes.

- -trendable

  (optional) Specifies a test object for trending. Must be used with -name and -value.

  Refer to "Use Scenario for Adding Attributes to a Trend UCDB" for information on usage flow and a detailed example.

- -ucdb

  Required only when used specifying attributes as global attributes. Specifies global UCDB object for attributes.

Examples

- Show all test records in a UCDB that has been loaded into coverage view mode:

```
coverage attribute -test *
```

- Add a trendable attribute to a "snapshot" UCDB:

```
vsim –c –viewcov snapshot1.ucdb
coverage attribute –ucdb -name Tests -value 87 -trendable
coverage attribute –ucdb -name Fails -value 47 -trendable
coverage attribute –ucdb -name Passes -value 40 -trendable
coverage save snapshot1.ucdb
```

- Create a new covergroup, in an existing testplan, with type_option.merge_instances=1:

```
vsim -c -viewcov testplan.ucdb
coverage create -cvg mycvg -parent /top
coverage attribute -path /top/mycvg -name MERGEINSTANCES -value 1
coverage attribute -path /top/mycvg -name {#GOAL#} -value 90
```

- Display ALL attributes on the "top/mytest" history node:

```
coverage attribute -history top/mytest
```

- Assign (or overwrite) the value "george" on the "responsible" attribute in the "merge1" history node:

```
coverage attribute -history merge1 -name responsible -value george
```

- Display ALL attributes on the "merge1" history node, within a specified UCDB:

```
coverage attribute -history merge1 my.ucdb
```

Related Topics

- Verification Management Overview
- Verification Management Browser Window
- Verification Tracker Window
- Understanding Stored Test Data in the UCDB
- coverage exclude, coverage goal
- coverage analyze
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage testnames
- coverage weight
- coverage unlinked
- vcover attribute
- vcover merge
- vcover ranktest
- vcover history
- vcover stats

coverage clear

The coverage clear command clears specified types of coverage data from the coverage database.

When entered at the simulation prompt (simulation mode), performing coverage clear on an instance affects the code coverage data of the associated design unit. The reverse is also true, that if you perform “coverage clear” on a design unit, the associated instances of that design unit are also cleared.

However, when issued at the vsim prompt with the vsim -viewcov command (batch or post-processing modes), coverage clear does not synchronize code coverage data between instances and associated design units. So, clearing an instance has no effect on code coverage data for associated design units. Conversely, clearing a design unit has no affect on related instances.

Syntax

coverage clear [-code {b | c | e | f | s | t |}...] [-codeAll]
[-assert] [-cvg] [-directive]
[-du <du_name> | -instance <pathname>] [-path <obj>+] [-match <string>] [-recursive]

Arguments

- -assert

  (optional) Clears assertion data only.

- -code {b | c | e | f | s | t |}...

  (optional) Clears code coverage data for coverage type: b=branch coverage; c=condition coverage; e=expression coverage; s=statement coverage; t=toggle; f=Finite State Machine coverage. More than one of the coverage types may be specified with a single argument.

- -codeAll

  (optional) Specifies the command apply to all coverage types. Equivalent to -code bcestf.

- -cvg

  (optional) Clears covergroup data only.

- -directive

  (optional) Clears cover directive data only.

- -du <du_name>

  (optional) Specifies design unit to clear of specified types of coverage data. To specify all design units in the current dataset, specify <du_name> as “*”.

- -instance <pathname>

  (optional) Clears the specified coverage data for the specified instances; -instance can be specified multiple times.

- -match <string>

  (optional) Recursively matches the given pattern against the specified coverage types in the entire instance tree. If -duname is specified, it matches against the specified coverage types in the design unit. Wildcards are allowed. This switch is mutually exclusive with -recursive.

- -path <obj>+

  (optional) Specifies that the subtrees being cleared are rooted at the specified design node. Multiple objects may be specified. The <obj> can be used to specify a dataset other than the current dataset. (Refer to "Object Name Syntax" for instructions on how to specify a dataset.) If no dataset is specified, the current dataset is used. Only one dataset name per command invocation may be used or an error will result. This switch applies to a sub-hierarchy.

- -recursive

  (optional) Specifies that the command is applied recursively. Optional. The default is for the query to be restricted to the single object or objects specified in the command. This switch is mutually exclusive with -match.

Examples

- coverage clear

  Clears all coverage data from the current simulation database (UCDB).

- coverage clear -cvg -directive

  Clears data for all covergroups and covergroup directives.

- coverage clear -path /top/a/*

  Clears coverage data from all */top/a*.

Related Topics

- Code Coverage
- coverage attribute
- coverage exclude
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage analyze

coverage create

The coverage create command creates covergroups, coverpoints, crosses and bins under a specified parent in an existing UCDB file. It can be used only in Coverage View mode (vsim -viewcov...) as a method of adding these functional coverage items to existing UCDBs. Once created, the covergroup items behave like any SystemVerilog covergroup items, and coverage calculation follows SystemVerilog coverage calculation rules.

This command allows you to make changes to your hierarchy. Backups are recommended.

Syntax

To create a covergroup, covergroup instance or coverpoint:

```
coverage create (-cvg | -cvinstance | -cvp) <name> -parent <path>
```

To create a cross:

```
coverage create -cross <name> -parent <path>
                -item <cvpname1> -item <cvpname2> [-item <cvpnameN>
```

To create a covergroup bin:

```
coverage create -cvb <name> -parent <path> [-count <count_val>] [-illegal | -ignore]
```

Arguments

- -cvb

  (Required for bin creation) Specifies creation of a bin of specified unique <name> under the specified parent scope. Parent specified must be a coverpoint or cross, and must not be a TYPE item (for example, under covergroup Type scope) with merge_instances = 0. By default, a regular covergroup bin is created.

- -cvg

  (Required for covergroup creation) Specifies creation of a covergroup. Must specify the <name> and the parent <path> for the covergroup being created.

- -cvinstance

  (Required for covergroup instance creation) Specifies creation of a covergroup instance, such as a covergroup instance. Must specify a unique <name> and a parent <path>.

- -cvp

  (Required for coverpoint creation) Specifies creation of a coverpoint. Must specify a unique <name> and parent <path>. Parent scope must be covergroup or covergroup instance.

- -cross <name>

  (Required for cross creation) Specifies creation of a cross. Must specify a unique <name> and a parent <path>. Parent scope must be covergroup or covergroup instance and must have at least two -item arguments specified.

- -ignore

  (optional) Used only with -cvb to create an ignore bin.

- -illegal

  (optional) Used only with -cvb to create an illegal bin.

- -item <cvpname1> -item <cvpname2> [-item <cvpnameN>

  (optional) Creates the specified new instance scope(s) to the context tree, at the specified <path>. The <path> specified must be absolute, not relative. May be specified multiple times per coverage create command.

- <name>

  (required) Specifies a name for the specified covergroup, coverpoint, cross, or bin which is unique within the parent scope.

- -parent <path>

  (required) Specifies the path to the parent item of the specified covergroup, coverpoint. or cross. <path> specified must be absolute.

Examples

- Create a covergroup with *type_option.merge_instances*=1, and a goal of 90:

```
vsim -c -viewcov testplan1.ucdb
coverage create -cvg mycvg -parent /top
coverage attribute -path /top/mycvg -name MERGEINSTANCES -value 1
coverage attribute -path /top/mycvg -name {#GOAL#} -value 90
```

- Create a coverpoint with some bins:

```
vsim -c -viewcov testplan1.ucdb
coverage create -cvp p1 -parent /top/mycvg
coverage create -cvb b1 -parent /top/mycvg/p1 -count 7
coverage create -cvb b2 -parent /top/mycvg/p1
coverage create -cvb ii1 -parent /top/mycvg/p1 -ignore
coverage create -cvb il1 -parent /top/mycvg/p1 -illegal -count 5
```

- Create another coverpoint with some bins

```
vsim -c -viewcov testplan1.ucdb
coverage create -cvp p2 -parent /top/mycvg
coverage create -cvb {b[0]} -parent /top/mycvg/p2
coverage create -cvb {b[1]} -parent /top/mycvg/p2
coverage create -cvb i1 -parent /top/mycvg/p2 -ignore
coverage create -cvb l1 -parent /top/mycvg/p2 -illegal
```

- Update some bin counts:

```
vsim -c -viewcov testplan1.ucdb
coverage edit -setcount 3 -path {/top/mycvg/p2/b[0]}
coverage edit -incrcount 4 -path {/top/mycvg/p2/b[0]}
coverage edit -incrcount 9 -path {/top/mycvg/p2/b[1]}
coverage edit -incrcount 2 -path /top/mycvg/p2/i1
```

Related Topics

- Code Coverage
- coverage attribute
- coverage exclude
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage analyze

coverage edit

The coverage edit command opens a coverage dataset (*.ucdb*) to edit the contents. Used only in Coverage View mode (vsim -viewcov...). Use this command to alter existing UCDBs.

The strip and install versions of the command are perhaps most useful for merging a unit test module and a system test module.

Caution

This command allows you to make any change to your hierarchy you wish. Be aware that some changes in hierarchy at best may not make sense, and at worst will render your data unusable by the tools. Backups are recommended.

Syntax

To prune away objects from the UCDB:

coverage edit -keeponly | -delete
{-path <path> | -plansection <path> | -instofdu <du_name> | -test <test_name> [<coverage_types>]**}**...

where <coverage_types> are:

([-code {b | c | e | f | s | t |}...] [-codeAll] [-cvg] [-directive] [-assert])

To change or move around levels/hierarchies within the UCDB, use one of the following:

coverage edit -stripdesign <int> | -installdesign <path> ...

coverage edit -stripplan <int> | -installplan <path> ...

To move instances of a design or trees/subtrees/instances of a testplan, use :

coverage edit -movedesign <source path> <dest path>

coverage edit -moveplan <source path> <dest path>

To rename trees, instances of design units, design units, testplan sections, or tests:

coverage edit -rename <new_name> {-path <path> | -instofdu <du_name> | -du <path> | -plansection <path> | -test <test_name>}

To rename libraries, or sub-sections of libraries:

coverage edit -rename <new_name> -lib <old_libname> [-path <path> | -instofdu <du_name>]

To edit the paths for HDL source code, enabling the later merging of two UCDB's which are otherwise identical or structurally compatible:

coverage edit -rename -srcfilestring <target_substring> <replacement_substring>

To edit the count value of a particular bin:

coverage edit {-incrcount <count> |-setcount <count>} -path <path>

Arguments

- -assert

  (optional) Specifies all assertions to be acted upon.

- -code {b | c | e | f | s | t}...

  (optional) Specifies code coverage data to be acted upon for coverage type: b=branch coverage; c=condition coverage; e=expression coverage; s=statement coverage; t=toggle; f=Finite State Machine coverage. More than one of the coverage types may be specified with a single argument.

- -codeAll

  (optional) Edits all coverage types. Equivalent to -code bcestf.

- -cvg

  (optional) Edits covergroup data only.

- -delete

  (optional) Deletes specified UCDB subset or coverage types. Mutually exclusive with -keeponly. Not valid with -du. Accepts wildcards in paths specified.

- -directive

  (optional) Specifies that the command apply cover directive data only.

- -du <path>

  (optional) Edits the specified design unit. Used with the -rename option. Mutually exclusive with -path, -plan, -instofdu, and -test.

- -incrcount <count>

  (required to increment bin counts) Increments the count of the specified bin by the specified count value. The <path> specified must match with a valid bin (coveritem).

- -installdesign <path>

  (optional) Creates the specified new instance scope(s) to the context tree, at the specified <path>. The <path> specified must be absolute, not relative. May be specified multiple times per coverage edit command.

- -installplan <path>

  (optional) Creates the specified new testplan scope(s) to the testplan tree, at the specified <path>. The <path> specified must be absolute, not relative. May be specified multiple times per coverage edit command.

- -instofdu <du_name>

  (optional) Edits all instances of specified design unit.

- -keeponly

  (optional) Keeps only given UCDB subset or coverage types. Mutually exclusive with -delete. Not valid with -du. Accepts wildcards in paths specified.

- -lib <old_libname>

  (optional) Valid only for use with the -rename argument. Specifies a particular library instance name to be replaced with the name specified by -rename.

- -movedesign <source path> <dest path>

  (optional) Moves a scope in the context tree from an existing location, specified by <source path>, to a new parent <dest path>. Accepts wildcards in paths specified.

- -moveplan <source path> <dest path>

  (optional) Moves a scope in the testplan tree from an existing location, specified by <source path>, to a new parent <dest path>. Accepts wildcards in paths specified.

- -path <path>

  (optional) Edits the specified design or testbench hierarchy. Mutually exclusive with -plan, -instofdu, and all <coverage types>.

  When used with -incrcount or -setcount, <path> must be to a valid bin (coveritem).

- -plansection <path>

  (optional) Edits the specified testplan section. Mutually exclusive with -path, -instofdu, and -test.

- -rename <new_name>

  (optional) Renames a single scope of a path to the specified name. <new_name> is not a full path, but simply the scope to be renamed (wildcards are accepted). The hierarchical path to the target scope to be renamed must be identified with the -path <path> argument.

- -setcount <count>

  (required to set bin count) Sets the count of the specified bin to the specified count <value>, which must be a positive integer. Must be specified with a <path>, and the path specified must be to a valid bin (coveritem).

- -srcfilestring <target_substring> <replacement_substring>

  (optional) Used with -rename to replace a specific string in the HDL source code path with different string. <target_substring> is the string to be searched; <replacement_substring> is the string which replaces the first matching string found in the path to the VHDL or Verilog source. Wildcards not supported. Resulting UCDB can be be merged with an identical or structurally compatible UCDB file.

- -stripdesign <int>

  (optional) Deletes the specified number of levels (<int>) of topmost design instance scopes in the context tree. When you strip a design, non-instance subscopes of that design are also removed. If levels specified are out of range, a warning is issued. May be specified multiple times per coverage edit command.

- -stripplan <int>

  (optional) Deletes the specified number of levels (<int>) of topmost testplan scopes in the testplan tree. If levels specified are out of range, a warning is issued. May be specified multiple times per coverage edit command.

- -test <test_name>

  Edits a specified test name. Mutually exclusive with -path, -plan, and -instofdu.

Examples

Note that in some examples listed below, proper escape methods must be used, as Coverage View mode does not support the use of certain characters.

- Keep only all instances of the given design unit:

```
coverage edit -keeponly -instofdu myentity(myarch)
```

- Discard statement coverage from the entire database (the -path / is assumed):

```
coverage edit -delete -code s
```

- Discard statement coverage from all instances (including all subtrees) of a DU in the given database:

```
coverage edit -delete -code s -instofdu mydu
```

- Keep only coverage from a specified testplan section:

```
coverage edit -keeponly -plansection /myplan/1.2
```

- Keep only coverage from a specified design unit:

```
coverage edit -keeponly -instofdu /top/a
```

- Rename only a testplan item */top/Specific Code Coverage*

```
coverage edit -plan {/top/Specific Code Coverage} -rename new_name
```

- Rename design unit *work.statemach*

```
coverage edit -du work.statemach -rename new_name
```

- Rename test *sink_b* to *sink_b2*

```
coverage edit -test sink_b -rename sink_b2
```

- Change the library name part of "work" design units to “libA”

```
coverage edit –rename –lib work libA
```

- Change the library name part of all design units to “libA”

```
coverage edit –rename –lib {} libA
```

- Remove the library name part of "work" design units

```
coverage edit -rename -lib work {}
```

- Change the library name part of "work.v*" design units to “libA”

```
coverage edit -rename -du *.v* -lib work libA
```

- Move testplan scope "b" and its entire subtree so that "b" is now a child of "testplan":

```
coverage edit -moveplan /testplan/a/b /testplan
```

- Move a group of toggles:

```
coverage edit -movedesign /top/fib_val /top/mach
```

- Move one toggle of many

```
coverage edit -movedesign /top/st2[0] /top/mach
```

- Move a many statements:

```
coverage edit -movedesign /top/machinv/branch* /top
```

- Move one expression to a newly created path of dummy instance scopes:

```
coverage edit -installdesign /tic/tac/toe
coverage edit -movedesign /top/fib/expr#72# /tic/tac/toe
```

- Strip top 2 levels of the UCDB being viewed and replace it with */foo/bar:*

```
coverage edit -stripdesign 2 -installdesign /foo/bar
```

Related Topics

- Coverage View Mode and the UCDB
- Coverage and Verification Management in the UCDB
- Verification Management Browser Window
- Verification Tracker Window
- coverage analyze
- coverage attribute
- coverage exclude
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage weight
- vsim -viewcov option

coverage exclude

The coverage exclude command allows you to exclude the following from coverage statistics:

- specific code coverage items (statement, branch, expression or condition)
- specific code coverage types
- all code in specified source file(s)
- lines within a source file
- specific items on a line within a source file
- rows within a condition or expression truth table
- code inside specific design units or instances
- transitions or states within a Finite State Machine
- toggle nodes
- individual transitions of a toggle node
- assertions
- cover directives
- functional coverage (covergroups)

This command and its arguments (except -comment) can be issued during simulation or in Coverage View (post-process) mode.

Refer to "Methods for Excluding Objects" and "Excluding Functional Coverage from the GUI and Reports" for details on the different methods for excluding code and functional coverage, as well as the advantages of particular methods.

You can add/edit comments for an exclusion using the -comment option, which is supported for use in Coverage View mode only.

Coverage exclusions can be removed using the -clear switch.

Wildcard usage is supported. See specific switches for usage information and the Examples section for example commands.

File based exclusions cannot be cleared by scope. For example, an exclusion that was set using -srcfile cannot be cleared later using -scope.

Syntax

Comment-related commands for exclusions (only available in Coverage View mode):

```
coverage exclude <exclusion> -comment <string>
```

For file-based, line-based, or wholesale exclusions of coverage:

```
coverage exclude
    {   -srcfile <source_file_list> [-pragma] |
        -du <du_name> [-srcfile <source_file_list>] [-pragma] |
        -scope <scope_path> [ -srcfile <source_file_list> | -recursive ] }
    [-linerange [<ln>] ... [<ln>-<ln>] ...] [-item {<bces>} [[<int> | <int-int>] ...]]
    [-allfalse] [-dataset <name>] [-code {a | b | c | d | e | f | s | t}...] [-clear]
```

To exclude expression or condition rows:

```
coverage exclude
    {   -srcfile <source_file_list> [-pragma] |
        -du <du_name> [-srcfile <source_file_list>] [-pragma] |
        -scope <scope_path> [-srcfile <source_file_list>] }
    [-udpcondrow <ln> [<rn>] ... [<rn>-<rn>] ...]
    [-udpexprrow <ln> [<rn>] ... [<rn>-<rn>] ...]
    [-item {<bces>} [[<int> | <int-int>] ...]]
    [-feccondrow <ln> [<rn>] ... [<rn>-<rn>] ...]
    [-fecexprrow <ln> [<rn>] ... [<rn>-<rn>] ...]
    [-dataset <name>] [-clear]
```

To exclude functional coverage (assertions, cover directives, or covergroups), name the objects to exclude. Wildcard characters are allowed when running the command in Coverage View mode.

```
coverage exclude
    { -assertpath <assert_list> | -dirpath <directive_list> | -cvgpath <cvg_list> }
    [-du <du_name> | -srcfile <source_file_list>]
    [-clear] [-comment <string>]
```

To exclude assertions and/or cover directives by line range:

```
coverage exclude
    { -srcfile <source_file_list> |
    -du <du_name> [ -srcfile <source_file_list> |
    -scope <scope_path> [-srcfile <source_file_list> | -recursive ] }
    [-linerange [<ln>] ... [<ln>-<ln>] ...] [-comment <string>]
    [-code {a | b | c | d | e | f | s | t}...] [-immediate] [-concurrent]
    [-dataset <name>] [-clear] [-lang sv | psl | vhdl]
    [-severity note | warning | error | fatal]
```

Immediate and concurrent assertions/cover directives are specified with the -immediate and -concurrent options, respectively. If neither -immediate nor -concurrent are specified, both types are excluded.

If -code is not specified, the exclusion will apply to assertions and cover directives in addition to code coverage types (equivalent to -code abcdefst.)

To exclude FSM states or state transitions:

```
coverage exclude
    ( -du <du_name> {<fsm_args> | -pragma | -scope <scope_path> } |
    [-dataset <name>]) [-clear]
```

where <fsm_args> are:

```
-fstate <state_var_name> [<state_name>] ... }
-ftrans <state_var_name> [<transition_name>] ... |
-fsamestate <state_var_name> [<state_list>] }
-fcontains <state_var_name> [<two_state_trans_list>] }
-finitial <state_var_name> [<state_list>] }
[-dataset <name>] [-clear]
```

By way of illustrating the above syntax, consider the following syntax examples.

To exclude FSM transitions starting with specific states:

```
coverage exclude -du <du_name> -finitial <state_var_name> [<state_list>]
```

To exclude FSM multi-state transitions containing specific 2-state transitions:

```
coverage exclude -du <du_name> -fcontains <state_var_name> [<two_state_trans_list>]
```

To exclude FSM same state transitions from coverage:

```
coverage exclude -du <du_name> -fsamestate <state_var_name> [<state_list>]
```

To exclude an entire state machine from coverage:

- if auto exclusions are enabled:

  ```
  coverage exclude -fstate <state_var_name>
  ```

- if auto exclusions are not enabled:

  ```
  coverage exclude -fstate <state_var_name> -ftrans <state_var_name>
  ```

To exclude toggle coverage:

```
coverage exclude
    -togglenode <node_path_list> [-du <du_name> | -scope <scope_path> [-recursive]]
    [-dataset <name>] [-in] [-out] [-inout] [-internal] [-ports] [-clear] [-pragma]
    [-trans <transition_list>]
```

Arguments

- -allfalse

  (optional) Modifies branch exclusion algorithm by applying exclusions to the false path of a branch when the branch does not have an explicit "else". This argument applies to branch coverage only and includes the all false branch in case statements. Branch coverage (on by default) must be turned on for this argument to take effect. The line number(s) specified with the -linerange argument, if used, must include the line on which the if-branch appears. For more information about allfalse and if-else branches, refer to “Branch Coverage”. See also Exclude AllFalse Branches in Case Statements.

- -assertpath <assert_list>

  (optional) Excludes the assertion with the specified path from coverage. Wildcard usage is allowed in Coverage View mode only.

- -code {a | b | c | d | e | f | s | t}...

  (optional) Excludes coverage objects of the specified type from the specified dataset. a=assertions; b=branch coverage; c=condition coverage; d=cover directives; e=expression coverage; f=Finite State Machine coverage; s=statement coverage; t=toggle coverage (either regular or extended). If -code is not specified, all assertions and cover directives as well as all possible coverage types are excluded. More than one coverage can be specified with each -code argument. If -item is used, -code f or t is not valid.

- -clear

  (optional) Removes exclusions from dataset. The “coverage exclude” command operates so as to add exclusions whenever -clear is not specified.

- -comment <string>

  (optional) Attaches to the excluded item a comment string describing the exclusion. Valid in post-processing (Coverage View) mode only. The comments appear in the coverage exclusions reports created by coverage report and vcover report, and in the Source window of the GUI by hovering the cursor over the line in the Hits column. Clearing an exclusion clears any associated comment. Comments can be added, modified and removed either from the GUI, or by reapplying the same command used initially, replacing the <string> with a new comment. The comment can be removed by using an empty <string> for the comment.

- -concurrent

  (optional) Excludes concurrent assertions/cover directives only.

- -udpcondrow <ln> [<rn>] ... [<rn>-<rn>] ...

  (optional) Specifies condition truth table row(s) <*rn*> in the specified line <*ln*> to be excluded from coverage. Multiple rows, or ranges of rows, separated by spaces, are allowed. If no row number is specified, all rows are excluded.

- -cvgpath <cvg_list>

  (optional) Excludes the covergroup with the specified path from coverage. Wildcard usage is allowed in Coverage View mode only.

- -dataset <name>

  (optional) Specifies dataset into which exclusions are to be applied. Only one dataset name per command invocation may be used or an error will result. If not specified, the current dataset is assumed ("sim" is the default when running interactively). All specified objects, such as scopes, design units, or variable names, must be present in the named dataset. (Refer to “Object Name Syntax” for instructions on how to specify a dataset.)

- -dirpath <directive_list>

  (optional) Excludes the cover directive with the specified path from coverage. Wildcard usage is allowed in Coverage View mode only.

- -du <du_name>

  (required only for du-based exclusions) Specifies design unit to be excluded. Multiple -du specifications are allowed. Mutually exclusive with -scope, which is the default. The <du_name> must include the path to the design unit being excluded. Wildcards are supported. For example, you can specify all design units in the current dataset with the "*" wildcard – for example, coverage exclude -du *.

- -udpexprrow <ln> [<rn>] ... [<rn>-<rn>] ...

  (optional) Specifies expression truth table row(s) <*rn*> in the specified line <*ln*> to be excluded from coverage. Multiple rows, or ranges of rows, separated by spaces, are allowed. If no row number is specified, all rows are excluded.

- -fcontains <state_var_name> [<two_state_trans_list>]

  (optional; must be used with -du fsm) Excludes all FSM multi-state transitions (more than two-state) from coverage (for the specified Finite State Machine) which contain any of the transition(s) specified within the <two_state_trans_list>. For example, the command: “coverage exclude -du fsm1 -fcontains s0>s1” excludes all multi-state transitions in *fsm1* which contain state changes from s0>s1.

- -feccondrow <ln> [<rn>] ... [<rn>-<rn>] ...

  (optional) Excludes specified row in focused expression coverage (FEC) condition coverage with a specified line number from the report. Multiple rows, or ranges of rows, separated by spaces, are allowed. If no row number is specified, all rows are excluded.

- -fecexprrow <ln> [<rn>] ... [<rn>-<rn>] ...

  (optional) Excludes specified row in focused expression coverage (FEC) expression coverage with a specified line number from the report. Multiple rows, or ranges of rows, separated by spaces, are allowed. If no row number is specified, all rows are excluded.

- -finitial <state_var_name> [<state_list>]

  (optional; must be used with -du fsm) Excludes all Finite State Machine (FSM) state transitions from coverage (for the specified FSM) which start with a state within the <state_list>. For example, the command: “coverage exclude -du fsm1 -finitial cst s0” excludes all transitions in *fsm1* with the starting state of s0. Optionally, you can specify a list of states (state_list) to which the exclusion is applied.

- -fsamestate <state_var_name> [<state_list>]

  (optional; must be used with -du fsm) Excludes all Finite State Machine (FSM) same state transitions from coverage (for the specified FSM). For example, the command: “coverage exclude -du fsm1 -fsamestate cst” excludes all same state transitions in *fsm1*. Optionally, you can specify a list of states (s1 s2 s3) to which the exclusion is applied.

- -fstate <state_var_name> [<state_name>] ...

  (optional) Specifies the Finite State Machine state or states to be excluded from coverage for the specified FSM, specified with <state_var_name>. Multiple states, separated by white space, are allowed. If no state name is specified, all states are excluded. By default, when a

state is excluded, all transitions to and from the state are excluded. This behavior is called "auto exclusion". To explicitly control auto exclusion, set the vsim argument -autoexclusionsdisable to fsm or none. To change the default behavior of the tool, set the variable AutoExclusionsDisable in the *modelsim.ini* file.

- -ftrans <state_var_name> [<transition_name>] ...

  (optional) Specifies the transition states to be excluded for the specified FSM (state_var_name). <transition_name> is "<state_name>-><state_name>". Multiple transitions, separated by white space, are allowed. If no transition is specified, all transitions are excluded. If whitespace is present within the transition, it must be surrounded by curly braces.

- -immediate

  (optional) Excludes immediate assertions/cover directives only.

- -in

  (optional) Excludes the specified toggle nodes of mode IN. This argument is valid only when -togglenode is specified.

- -inout

  (optional) Excludes the specified toggle nodes of mode INOUT. This argument is valid only when
  -togglenode is specified.

- -internal

  (optional) Excludes the specified toggle nodes of internal (non-port) objects. This argument is valid only when -togglenode is specified.

- -item {<bces>} [[<int> | <int-int>] ...]

  (optional) Excludes specified coverage item(s) on a line of source code from database. The **-item** argument can only be applied to coverage exclude command entries for the line number specified with -linerange, -udpcondrow, or -udpexprrow. <bces> is required and is used to specify one or more of the coverage types to exclude: branch, condition, expression, and/or statement. Items are numbered in left to right order within a line, regardless of hierarchy, from 1 upward. Only one -item argument allowed with each coverage exclude command. This argument may not be used with the -code tf argument.

- -lang sv | psl | vhdl

  (optional) Specifies assertions/cover directives of a specific language (SV, PSL, or VHDL). You may specify multiple languages by using this option multiple times. If -lang is not specified, all three languages are selected.

- -linerange [<ln>] ... [<ln>-<ln>] ...

  (optional) Specifies the line number(s) and/or range of line numbers to be excluded from code coverage in the design source file *-srcfile <source_file>*. Multiple lines and line ranges are permitted, separated by whitespace.

  - Multi-line statements, branches, conditions or expressions are selected by specifying the last line number of that item.
  - When -linerange is not specified, all objects on all lines of the specified design unit, scope, or source file are excluded. This is referred to as a "wholesale exclusion".
  - -srcfile is required for -linerange unless -du or -scope is used, and only one source file is used to implement the du or scope.
  - If -srcfile is used together with -du/-scope, and -linerange is in effect, it is possible for -linerange to specify lines other than lines used to implement the -du or -scope. Such lines are ignored.

- -out

  (optional) Excludes the specified toggle nodes of mode OUT. This argument is valid only when -togglenode is specified.

- -ports

  (optional) Excludes the specified toggle nodes of mode IN, OUT, or INOUT. This argument is valid only when -togglenode is specified.

- -pragma

  (optional) Adds or clears pragma and user exclusions. Operates with file-based exclusions (-du and/or -srcfile) for all coverage types, including toggle exclusions (-togglenode). For instance based exclusions (-pragma used with -scope) it operates with all coverage types except FSM coverage. If the -pragma argument is specified, both user and pragma exclusions are applied. If the option is not specified, only user exclusions are applied.

- -recursive

  (optional) Used with -scope only. Specifies that exclusions apply recursively into subscopes. If omitted, the exclusions are limited to the current scope.

- -severity note | warning | error | fatal

  (optional) Specifies the assertion/cover directive severity level. When -severity is specified only assertions/cover directives with the same or lower severity are selected. If not specified, assertions/cover directives of all severities are selected.

- -scope <scope_path>

  (required only for instance-based exclusions) Specifies the scope to be excluded. Mutually exclusive with the -du argument, and -scope is the default if neither argument is used. Multiple -scope specifications are allowed. Wildcards are not allowed. To recursively exclude scopes, use with -r.

- -srcfile <source_file_list>

  (required only for file-based exclusions) Specifies source file to be excluded. Multiple file specifications are allowed, separated by white space.

- -togglenode <node_path_list>

  (optional) Specifies the named nodes for toggle exclusion. Multiple nodes separated by spaces are allowed. Part-selects are accepted — with node and selects surrounded by curly braces — using the following format:

  {toggle_node[<int>:<int>]} — matches an integer index within the range

  {toggle_node(<int> to <int>)} — matches an integer index within the range

  {toggle_node(<int> downto <int>)} — matches an integer index within the range

  By default, specified toggle nodes to be excluded are relative to the current scope. If used with -du, specified toggle nodes excluded are relative to the design unit.

- -trans <transition_list>

  (optional) Specifies the toggle node transition(s) to be excluded from coverage. Multiple transitions separated by spaces are allowed. Transition names are not case sensitive and can be any of the following six: 01 10 0Z 1Z Z0 Z1

Examples

- Recursively exclude branch coverage from instance */top/dut*.

```
coverage exclude -scope /top/dut -r -code b
```

- Exclude statement, else branch, expression and condition coverage from line 10 to 20 in file *project1.vhd*.

```
coverage exclude -srcfile project1.vhd -linerange 10-20
```

- Exclude statement, branch, expression, and condition coverage from instance */top/dut* in dataset *tt* from line 102 through 110 and line 200 through 250 in the source file *project1.vhd*.

```
coverage exclude -scope /top/dut -dataset tt -srcfile
   project1.vhd -linerange 102-110 200-250
```

- Remove statement, branch, expression, condition, and fsm exclusions from the source file *project1.vhd*.

```
coverage exclude -clear -srcfile project1.vhd
```

- Add rows 2 through 4 from the condition truth table on line 115 to the code coverage exclusions for source file *project1.vhd*.

```
coverage exclude -srcfile project1.vhd -udpcondrow 115 2-4
```

- Add all rows from the expression truth table on line 220 to the code coverage exclusions for source file *project1.vhd*.

```
coverage exclude -srcfile project1.vhd -udpexprrow 220
```

  or

```
coverage exclude -srcfile project1.vhd -linerange 220 -code e
```

- Exclude the all false (00) branch of the following case:

```
else if
   (A | B)
```

Assuming that leading "if" is on line 100 of file (file f.v), enter this command:

```
coverage exclude -srcfile f.v -linerange 100 -allfalse
```

- Exclude transitions S1->S2 and S2->S0 for FSM state in instance */top/dut/fsm1*.

```
coverage exclude -scope /top/dut/fsm1 -ftrans state S1->S2 S2->S0
```

- Exclude state S1 for FSM state in the design unit "fsm". If auto exclusions are on, all transitions to and from S1 are also excluded.

```
coverage exclude -du fsm -fstate state S1
```

- Exclude all cover groups in a module in Coverage View mode:

```
coverage exclude -du <module_name> -cvgpath *
```

or

```
coverage exclude -src <file_name> -cvgpath *
```

- Exclude all cover groups in a module in live simulation mode:

```
coverage exclude -du <module_name> -cvgpath <cvg_type1> <cvg_type2>
```

or

```
coverage exclude -src <file_name> -cvgpath <cvg_type1> <cvg_type2>
```

- Exclude from coverage all design units that begin with the prefix “assert_”:

```
coverage exclude -du assert_*
```

- Remove user and pragma exclusions for all toggle coverage. This is equivalent to 'toggle enable -all'.

```
coverage exclude -du * -code t -clear -pragma
```

- Exclude all toggle coverage (equivalent to 'toggle disable -all')

```
coverage exclude -du * -code t -pragma
```

- Exclude toggle nodes a, b, and c in instance */top/dut*.

```
coverage exclude -togglenode a b c -scope /top/dut
```

- Recursively exclude all input toggle nodes in instance */top/dut*.

```
coverage exclude -togglenode * -scope /top/dut -in -r
```

- Exclude bits of a bus from toggle nodes in instance */top/dut/bus_int*.

```
coverage exclude -togglenode {MyBus(3 downto 2)} -scope
\/top/dut/bus_int
```

- Exclude transitions 0->1 and 0->Z in toggle nodes *mybit* and *myreg*.

```
coverage exclude -togglenode mybit myreg -trans 01 0z
```

- Exclude covergroup at */top/tb/pci_cg*:

```
coverage exclude -cvgpath /top/tb/pci_cg
```

What NOT to do: Illegal Examples

```
coverage exclude -srcfile project1.vhd -code s -allfalse
```

- -allfalse has no effect because branch coverage is not specified.

```
coverage exclude -srcfile project1.vhd -linerange 10-20 -code ft
```

- There is no file name and line number associated with FSM and toggle coverage.

```
coverage exclude -scope /top/dut -srcfile project1.vhd -linerange 10-20 -r
```

- -r does not work with -srcfile or -linerange

```
coverage exclude -du * -srcfile project1.vhd -linerange 10-20
```

- '-du *' does not work with -srcfile or -linerange

```
coverage exclude -scope /top/dut -srcfile project1.vhd -line 10-20 -pragma
```

- -pragma does not work with -scope for FSM coverage

Related Topics

- Code Coverage
- Coverage Exclusions
- Coverage and Verification Management in the UCDB
- "Verification Management Browser Window
- "Verification Tracker Window
- coverage analyze
- coverage report
- coverage save
- "Toggle Coverage
- toggle add
- toggle enable
- toggle disable

coverage goal

The coverage goal command sets or displays the values for three separate kinds of goals, only one of which affects the total coverage calculation:

- Bin goal (for example, at_least) — sets the goal for a specific bin or coverage item within the coverage database (UCDB), which may change the total coverage calculation for that bin or item, as well as any coverage calculation in which that bin or item participates.
- Scope goal — sets the goal for specific scopes in the design hierarchy. It has no effect on coverage numbers. Instead, it affects the numbers that appear in the "% of Goal" columns in various coverage windows.
- Summary goal — sets the value of UCDB-wide goals for different coverage types. Has no effect on coverage numbers. There are two kinds of summaries: summary by design unit and summary by instance.

Goals are displayed by entering a coverage goal command without an <integer_weight>.

For more information regarding type coverage statistics gathered, refer to "Calculation of Total Coverage".

Syntax

Setting goals for individual coverage items (bins)

coverage goal -bins -path <path> <goal_number>

Setting goals for scopes, either in the design or testplan hierarchy

coverage goal {-du <du_name> | -path <path> | -plansection <section_name>} <goal_number>

Setting goals for database-wide summary by type

coverage goal
[-code {[b] [c] [e] [f] [s] [t]}] [-codeAll] [-fstate] [-ftrans] [-cvg] [-cvp] [-directive] [-assert] [-active] [-attempted] [-disabled] [-fail] [-pass] [-vpass] [-bydu] [-byinstance] [-feccond <cond>] [-fecexpr <expr>] [-type] [-udpcond <cond> [-udpexpr <expr>] <goal_number>

To display a currently set goal for any of the above, specify the command without **<goal_number>**.

Arguments

- -assert

  (optional) Sets goal for all assertion coverage sub-types.
- Set goal for assertion directive sub-types; arguments are:

  -active

  (optional) Sets goal for 'active' assertion directive, per instance.

-attempted

(optional) Sets goal for ‘attempted’ assertion directives, per instance.

-disabled

(optional) Sets goal for ‘disabled’ assertion directives, per instance.

-fail

(optional) Sets goal for assertion directive ‘failure’, per instance.

-pass

(optional) Sets goal for assertion directive ‘pass’, per instance.

-vpass

(optional) Sets goal for assertion directive vacuous passes, per instance.

Note

Only the use of -fail and -pass have an effect in the simulator because all other assertion sub-types do not contribute to coverage calculations. However, all types are stored in the database.

- -bins

  (optional) Used only with -path. Allows you to specify the “at_least” value for functional coverage bins (covergroup and cover directive). This switch is optional, unless a -path specification uses a wildcard. If it does, -bins is required to ensure that bins are matched. Otherwise, it matches functional coverage scopes only.

- -bydu

  (optional) Modifier used to set a per-du goal (code coverage only).

- -byinstance

  (optional) Modifier used to set a per-instance goal (code coverage and covergroup).

- -code {[b] [c] [e] [f] [s] [t]}

  (optional) Sets goal for specified code coverage types: b=branch coverage; c=condition coverage; e=expression coverage; s=statement coverage; t=toggle; f=Finite State Machine coverage. More than one coverage type can be (and at least one type must be) specified with each -code argument.

- -codeAll

  (optional) Sets or displays the goal for all code coverage types. Equivalent to -code bcestf.

- -cvg

  (optional) Sets goal for covergroup coverage type, summarizing by both type and instance.

- -cvp

  (optional) Sets goal for coverpoints and crosses, per-instance.

- -directive

  (optional) Sets goal for cover directive coverage type only.

- -du <du_name>

  (optional) Sets goal for a given design unit. Mutually exclusive with -path and -plansection. Cannot be combined with any other arguments besides -precision or <float percentage>.

- -feccond <cond>]

  (optional) Sets goal for the specified condition in focused expression coverage (FEC).

- -fecexpr <expr>

  (optional) Sets goal for the specified expression in focused expression coverage (FEC).

- -fstate

  (optional) Sets goal for FSM state coverage only.

- -ftrans

  (optional) Sets goal for FSM transition coverage only.

- <goal_number>

  (required in order to **set** goals; displays goals if left unspecified)

  For bin goals, <goal_number> is an integer.

  For scope or summary by type goals, <number> is a positive floating percentage between 0 and 100, depending on the granularity specified by -precision.

- -path <path>

  (optional) Sets the goal for a given coverage/design object. Mutually exclusive with -plansection and -du. <path> can be used to specify a dataset other than the current dataset. If no dataset is specified, the current dataset is used. Only one dataset name per command invocation may be used or an error will result. Cannot be combined with any other arguments besides -bins, -precision, or <number>.

- -plansection <section_name>

  (optional) Sets the goal for a specified testplan item. Mutually exclusive with -path and -du. Cannot be combined with any other arguments besides -precision or <number>.

- -precision <int>

  (optional) Precision for goal percentage. Default is 1 decimal place.

- -udpcond <cond>

  (optional) Sets goal for UDP condition.

- -udpexpr <expr>

  (optional) Sets goal for UDP expression.

- -type

  (optional) Sets goal for covergroup type coverage. The -type argument summarizes by type only, whereas the -cvg argument summarizes both by type and by instance.

Related Topics

- Code Coverage
- Coverage and Verification Management in the UCDB
- Verification Management Browser Window
- Timestamps and UCDB Modification or Merging
- Verification Management Overview
- Verification Tracker Window
- coverage analyze
- coverage attribute
- coverage exclude
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage weight

coverage loadtestassoc

Coverage test association data is not loaded by default when a test associated merged UCDB file is loaded in the GUI in Coverage View mode, unless the number of tests in the UCDB are less than 100. When the UCDB contains more than 100 tests, the test-associated data must be loaded explicitly using the coverage loadtestassoc command before using any of the test association features (coverage analysis).

Syntax

coverage loadtestassoc

Arguments

None

Examples

- Open the dataset file *test.ucdb*, load the test associated data, rank the tests in the UCDB, and determine which tests contribute the most coverage:

```
vsim -c -viewcov test.ucdb
coverage loadtestassoc
coverage ranktest
coverage analyze -coverage most
```

Related Topics

- Coverage View Mode and the UCDB
- Coverage and Verification Management in the UCDB
- Verification Management Browser Window
- Verification Tracker Window
- coverage analyze
- coverage attribute
- coverage ranktest
- coverage tag
- coverage weight
- vsim -viewcov option

coverage open

The coverage open command opens UCDB datasets for viewing in the GUI in Coverage View mode. Datasets can be closed once open using dataset close.

This command is equivalent to the command vsim -viewcov.

Syntax

coverage open <filename> [<logicalname>]

Arguments

- <filename>

  (required) Specifies the <filename>.ucdb to open in Coverage View mode. At least one UCDB is required.

- <logicalname>

  (optional) Specifies the logical name for the UCDB dataset. This is a prefix that will identify the dataset in the current session. By default the dataset prefix will be the name of the specified UCDB file.

Examples

- Open the dataset file *last.ucdb* and assigns it the logical name *test*.

```
coverage open last.ucdb test
```

Related Topics

- Coverage View Mode and the UCDB
- Coverage and Verification Management in the UCDB
- Verification Management Browser Window
- Verification Tracker Window
- coverage analyze
- coverage attribute
- coverage exclude
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage weight
- dataset close
- vsim -viewcov option

coverage ranktest

The coverage ranktest command ranks coverage data contained in the current Coverage View dataset (whether loaded with vsim -viewcov or in batch), according to each individual test's contribution to cumulative coverage. For any ranking operation performed on a merged result — the dataset must be from a merged UCDB, and must have been created with the vcover merge -testassociated argument.

The output of this command consists of what is written to *stdout*, as well as two lists of tests (contributing tests in *ranktest.contrib* and non-contributing tests in *ranktest.noncontrib*). The order of the tests written to stdout or a specified file (with -log) are listed in the order shown in Table 2-4:

Table 2-4. Order and Type of Ranked Tests

Contributing, compulsory	Mandatory tests, tests which need to be run regardless of achieved coverage	Sorted by total coverage %
Contributing, noncompulsory	Tests providing coverage not provided by any previous test.	Sorted by total coverage %
Non-contributing	Redundant tests, providing no incremental coverage.	Not sorted

To rank on a specific coverage item, design unit, or testplan section within the hierarchy of your design, use -path, -du or -plansection.

Syntax

vcover ranktest [<rankest_options>]
{<ucdb_or_coverstore_testN1> <ucdb_or_coverstore_testN2> ...}

<ranktest_options> =
[-algorithm {greedy | quick}]
[-assertion] [-assertfailure] [-code {b | c | e | f | s | t}...] [-codeAll] [-cvg] [-directive]
[-nocompulsoryordering | -compulsorysorting | -compulsoryranking]
[-error <msg_number>[,<msg_number>,…]]
[-fewest | -cputime | -simtime]
[-goal [<coverage_type>] <percentage>]
[-inputs <file_list>]
[-log <filename>]
[-maxcpu <real_num_in_seconds>] [-maxtests <int>]
[-metric {aggregate | total}]
[-note <msg_number>[,<msg_number>,…]]
[-path <path> | -du <du_name> | -plansection <path>] [-precision <int_num>]
[-quiet | -concise | -verbose]
[-rankfile <filename>]
[-suppress <msg_number>[,<msg_number>,…]]

[-warning <msg_number>[,<msg_number>,…]]
[-weight <coverage_type> <integer>]

Arguments

- -algorithm {greedy | quick}

  Specifies algorithm being used to produce the ranked list of tests: greedy is the default algorithm used. It produces the shortest list of tests contributing to coverage, arranged in order of highest incremental contribution to least. The quick algorithm provides an estimated ordering of tests, not guaranteed to be the shortest list or that the order of tests is perfectly determined as to a particular test's significance.

- -assertion

  (optional) Specifies ranktest for assertion data. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

  Traditionally, assertions are a measure of correctness and not a measure of coverage. When viewed as a coverage item, their definitions are as follows:

  - Percent coverage assertion passes (-assertion): Of all assertions what percent were evaluated to completion and had at least one non-vacuous pass.
  - Percent coverage assertion failures (-assertfailure): Of all assertions what percent were evaluated to completion and had at least one failure. This is THE ONLY NEGATIVE COVERAGE VALUE in the sense that 0% coverage is desirable and 100% coverage is very undesirable. As such, it is highly recommended that -assertfailure only be used in a mutually exclusive fashion with the other coverage values.

  It is possible for any given assertion to have both passes and failures. It is also possible for any given assertion to not ever be evaluated to completion. As such, often the sum of assertion passes and assertion failures will not be 100%.

- -assertfailure

  (optional) Specifies ranktest for failed assertion data only. Refer to "-assertion" for more information. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

- -code {b | c | e | f | s | t}...

  (optional) Specifies ranktest for corresponding code coverage type only: branch, condition, expression, FSM, statement, toggle. More than one coverage type may be specified with each -code argument (example: "-code bcest"). This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

- -codeAll

  (optional) Specifies ranktest for all coverage types. Equivalent to -code bcestf. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

- -nocompulsoryordering | -compulsorysorting | -compulsoryranking

  (optional) Specifies how the command ranks compulsory tests, when present.

  -nocompulsoryordering — (default) Compulsory tests are listed in a random order, with no ranking with respect to each other. Compulsory tests are not displayed in the detailed ranktest listing. In the GUI, the accumulated coverage data and incremental coverage data for compulsory tests is displayed as a minus sign (-). This option has the best performance of the three options.

  -compulsorysorting — Ranks compulsory tests in the order of descending 'total coverage' with respect to each other. Displays compulsory tests in the detailed ranktest listing according to this order. In the GUI, the accumulated coverage data and incremental coverage data for compulsory tests is calculated and displayed. This option has performance second to the -nocompulsoryodering option.

  -compulsoryranking — Ranks compulsory tests fully with respect to each other. Displays compulsory tests in the detailed ranktest listing according to this order. In the GUI, the accumulated coverage data and incremental coverage data for compulsory tests will be calculated and displayed. This option can have the worst performance of the three options.

- -concise

  (optional) Specifies the output is created with minimum additional I/O. This is the default output. Mutually exclusive with -quiet and -verbose.

- -cputime

  (optional) Specifies that the order of tests being selected for ranking is: the largest coverage gain per unit of CPU time. Mutually exclusive with the -fewest and -simtime arguments.

- -cvg

  (optional) Specifies ranktest for covergroup data only. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

- -directive

  (optional) Specifies ranktest for directive data only. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

- -du <du_name>

  (optional) Restricts the ranking to a specified design unit. This argument applies to a particular module type, by name, in all UCDB files. This option is mutually exclusive with -path and -plansection.

  Where <du_name> is [<library name>.]<primary>[(<secondary>)] and secondary name is required only for VHDL.

  <library name>. — (optional) Specifies the library name; if none is specified, then work is assumed. Must be followed by a period (.).

  <primary> — The name of the design unit.

  <secondary> — (optional) The secondary name of the design unit, required for VHDL.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -fewest

  (optional) Specifies that the tests be selected for ranking by: the largest coverage gain with the fewest number of tests. Mutually exclusive with the -cputime and -simtime arguments. Default.

- -inputs <file_list>

  (optional) Specifies a file containing ranktest arguments. The file can contain a list of UCDB files to be ranked.

- -goal [<coverage_type>] <percentage>]

  (optional) Specifies that the tests be ranked to achieve this coverage goal. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

  <coverage_type> — specifies that the goal is applied only to the specified type of coverage; otherwise, the goal is applied to all types of coverage.

  Valid <coverage_type> values:

  -assertfailure

  -cvg

  -directive,

  -code {b | c | e | f | s | t}...

  -codeAll.

  <percentage> — an integer value; the default is 100.

  The -goal argument can be specified multiple times, as shown in the following example:

  ```
  coverage ranktest -goal -assertion 90 -goal -code bcest 95
  ```

- -log <filename>

  (optional) Specifies the file for outputting ranked results. Output includes full path to tests. <filename> is a user specified string.

- -maxcpu <real_num_in_seconds>

  (optional) Monitors the accumulated CPU time of the ranked tests. Specifies the maximum CPU time to be allowed. If the specified number of seconds is exceeded, the ranking process is stopped.

<real_num_in_seconds> — Any integer greater than -1.0 where the default is -1.0 (no limit).

- -maxtests <int>

  (optional) Specifies threshold for the maximum number of tests to be ranked. When this threshold is exceeded, the ranking operation is terminated. <int> is any positive integer.

- -metric {aggregate | total}

  (optional) Indicates the kind of metric used for ranking. Arguments:

  total — (Default, unless any of the following arguments are used: -totals, -goal, -weight, -assertion, -assertfailures, -directive, -codeAll, or -code)

  total — (Default, unless any of the following arguments are used: -totals, -goal, -weight, -assertion, -assertfailures, -cvg, -directive, -codeAll, or -code)

  Ranking metric used produces values consistent with the totals for (verification) testplan sections obtained with coverage analyze.

  aggregate —

  Ranking metric used produces aggregate values based on each individual coverage type: values are *not* likely to be consistent with totals produced with coverage analyze.

  Each coverage type can be selected or not. Each coverage type can be given an individual weight and goal. These individual numbers are then combined and normalized to yield an aggregate metric that is unrelated to the number given by the total coverage.

  When the -metric aggregate argument is used, the resulting metric number will not “match” any other total coverage number produced by other verification tools (for example, coverage analyze).

  This is important because when you use any of the arguments (-totals, -goal, -weight, -assertion, -assertfailures, -cvg, -directive, -codeAll, or -code) with ranktest command, the aggregate metric is the default.

  Refer to “Merged Results vs Rank Report Coverage Why They Can Differ” for further details.

- -note <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "note." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information

- -path <path>

  Restricts ranking to design paths (non-testplan) matching the specified <path>. Optional. This argument can be used to rank on specific coverage items, or sub-trees. Mutually exclusive with -du and -plansection.

  <path> — Specifies either an absolute or relative path to the design files. On Windows systems the path separator should be a forward slash (/).

- -plansection <path>

  Restricts ranking to the specified testplan node. Optional. Mutually exclusive with -du and -path.

  <path> — Specifies either an absolute or relative path to the testplan. On Windows systems the path separator should be a forward slash (/).

- -precision <int_num>

  (optional) Specifies the decimal point precision for output. <int_num> is an integer value. The contents of the rank file are NOT affected by this argument. <int_num> is an integer value.

  <int_num> — Any positive integer where the default is 2.

- -quiet

  (optional) Creates the ranktest output without any additional I/O. Default creates ranktest with minimal I/O (-concise). Mutually exclusive with -concise and -verbose.

- -rankfile <filename>

  (optional) Specifies the name the ranktest file being created. Default if not specified is *ranktest.rank*. Can be specified with the vcover stats command to redisplay the results of this ranking. This file can also be used to repopulate the Browser with ranktest information.

  <filename> — A user specified string, which can include a path.

- -simtime

  (optional) Specifies that the order of tests being selected for ranking is: the largest coverage gain per unit of simulation time. Mutually exclusive to the -cputime and -fewest arguments.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the suppress variable in the *modelsim.ini* file to set a permanent default.

- <ucdb_or_coverstore_testN1> <ucdb_or_coverstore_testN2> ...

  (required unless the -inputs file is used) Specifies the name of two or more non-merged UCDB file(s) or Coverstores to rank, where <ucdb_or_coverstore_test> is: <ucdbfile>|<coverstore>[:<testname>].

- -verbose

  (optional) Specifies the output created with full I/O. Mutually exclusive with -quiet and -concise. Default is -concise.

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -weight <coverage_type> <integer>

  (optional) Used when selecting next ranking candidate. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

  <coverage_type> — specifies that the goal is applied only to the specified type of coverage; otherwise, the goal is applied to all types of coverage.

  Valid values for <coverage_type> are:

  -assertion

  -assertfailure

  -cvg

  -directive

  -code {b | c | e | f | s | t}...

  -codeAll.

  <integer> — Any non-negative integer.

  The -weight argument can be specified multiple times, as shown in this example:

```
coverage ranktest -weight -assertion 10 -weight -code bcest 5
```

Related Topics

- Code Coverage
- Coverage and Verification Management in the UCDB
- Verification Management Browser Window
- Verification Tracker Window
- coverage analyze
- coverage goal
- coverage tag
- coverage unlinked
- coverage weight
- vcover merge
- vcover ranktest
- vcover stats

coverage report

The coverage report command produces textual output of coverage statistics or exclusions. By default, the command prints results to the Transcript window, and returns an empty string. You can use the -file argument to save the output to a file.

You can choose from a number of report output options using the arguments listed below. You can access the coverage report functionality from the GUI through right-clicking in the Structure or Files windows and select **Code Coverage** > **Coverage Reports** from the popup context menu; or, **Tools > Code Coverage > Report**.

> **Tip**: HTML reports are created with the -html option, which is only compatible with the options listed therein. See the -html option for details.

By default, the command returns results from the current scope.

To specify a certain path for the report, you can use the -instance argument, or specify the specific cover directive or covergroup, such as:

- coverage report -instance <path>
- coverage report <path_to_cover_directive/covergroup>

> **Tip**: A report response of "No match" indicates that the report was empty. For example, "coverage report -du foo" where there is no design unit "foo" will result in "No match."

The command orders output on a by file basis unless you specify the -byinstance or -bydu argument.

To produce reports offline (for example, without a simulation loaded), use the vcover report command.

Syntax

coverage report [<coverage_arguments>]

Generic Coverage Arguments

coverage report [-bydu] [-above <percent> | -below <percent>] [-binrhs]
[-nocomment | -comment] [-details [-metricanalysis]] [-file <filename> [-append]]
[-flat [-primarykey <type | path | value> [-secondarykey <type | path | value>]]]
[-memory] [-noexcludedhits] [-nomissing] [-nozeroweights]
[-precision <int>] [-recursive [-depth <n>]]
[-setdefault [byfile | byinstance | bydu]] [-showambiguity]
[-testextract <test_name_or_pattern>] [-totals] [-xml] [-zeros]

Arguments for creating report in HTML

coverage report -html <input_ucdb> [-code [bcesf[t|x]] [-assert] [-binrhs] [-cvg] [-directive] [-details[=abcdefgst]] [-du <path/du>] [-hidecvginsts | -hidecvginstspi0] [-htmldir <outdir>]

[-instance <path/instance>] [-nodu] [-nofec] [-noudp] [-noframes] [-noinstance] [-notestplan][-notimestamps] [-nozeroweights] [-showexcluded] [-source] [-stmtaltflow] [-summary] [-testhitdata] [-testlist] [-threshL <val>] [-threshH <val>] [-usecnpm] [-verbose]

ONLY the above listed arguments are supported for use with HTML coverage reports.

Filtering Arguments - Selects one or more coverage types to appear in the report

coverage report [-assert] [-code {[b] [c] [e] [f] [s] [t]}...] [-codeAll] [-cvg] [-directive] [-failed] [-filter <pattern>] [-lang sva | psl | vhdl] [-severity info | note | warning | error | failure | fatal] [-testattr]

For each directive instance, the report includes by default the full instance path, the coverage count or percentage, the design unit, the source file name, and the source line number.

Code Coverage Arguments - Has no effect on assertion or functional coverage

coverage report [-byfile] [-byinstance] [-coverenhanced]
[-noannotate] [-nofec] [-noudp]
[-library <libname>] [-du <du_name>] [-package <pkgname>]
[-source <filename>] [-instance <path>] [-recursive [-depth <n>]]

For a Report of Non-collected Coverage Items

coverage report -nocollect [-file <filename> [-append]]

Exclusion Coverage Arguments

coverage report [-excluded [-pragma | -user] [-nocomment]]
[-code {[b] [c] [e] [f] [s] [t]}...] [-instance <path>] [-file <filename> [-append]]
[-showexcluded]

Toggle-specific Coverage Arguments

coverage report [-portmode (input | output | inout | internal)] [-verbose] [-all]

Toggle coverage statistics are available for reporting on instances, design units, and individual files. Toggle data is summed for all instances, and is reported by port or local name in the design unit, rather than by the connected signal. If you want toggle coverage statistics, you must specify either the -byinstance, -bydu, -instance <path>, or -du <du_name> arguments. If you do not use those arguments, or you use the -source <filename> argument, toggle coverage statistics are excluded even if you specify -code t. To get an itemized list of the signals, the -details argument is also required.

Assertion-specific Arguments

coverage report [-assert] [-lang sva | psl | vhdl] [-severity info | note | warning | error | failure | fatal] [-unattemptedimmed]

Cover directive-specific Arguments for SystemVerilog and PSL

coverage report [-config]

Covergroup-specific Arguments

coverage report [-covered] [-cvg] [-cvp] [-hidecvginsts | -hidecvginstspi0] [-nocvpbinsummary] [-nocrossbinsummary] [-nocvgbin] [-noignorebins] [-option] [-samples <spec>] [-usecnpm] [-uselimit]

Arguments

- -above <percent>

  (optional) Specifies that only objects with coverage values above this percentage be included in the output. <percent> is the coverage of coverpoints and crosses, not covergroups.

- -all

  (optional) When reporting toggles, creates a report that lists both toggled and untoggled signals. Counts of all enumeration values are reported. Not a valid option when reporting on a functional coverage database.

- -append

  (optional) Appends the current coverage statistics to the named output file (-file <filename>).

- -assert

  (optional) Reports only assertion coverage data.

- -below <percent>

  (optional) Specifies that only objects with coverage values below this percentage be included in the output. <percent> is the coverage of coverpoints and crosses, not covergroups. See also -above.

- -binrhs

  (optional) Specifies that for covergroups, a column is included in the report that displays the RHS for covergroup bins. The RHS is a sampled value that causes the bin to increment.

- -bydu

  (optional) Reports coverage statistics by design unit/module. The simulator will iterate through all design units in the design and report coverage data for each. Each design unit report will be the merged result of all instances of that design unit and will be sorted by design unit name. Can be made the default with the -setdefault bydu argument. You can also report coverage data for a specific design unit by using the -du <du_name> argument.

- -byfile

  (optional) Writes out a coverage summary for each source file in the design. This is the default report generated.

- -byinstance

  (optional) Writes out a coverage summary for all instances and packages. Can replace the default (-byfile) with the -setdefault byinstance argument. Can be used with the -recursive [-depth <n>] argument to report on all instances below the specified instance.

- -code {[b] [c] [e] [f] [s] [t]}...

  (optional) Specifies which code coverage statistics to include in the report. If -code is specified, the specification of one or more coverage types is required. By default, the report includes statistics for all categories you enabled at compile time. More than one coverage type can be specified with the -code argument.

  Not a valid option when reporting on functional coverage data.

  The coverage types allowed are as follows:

  b — Include branch statistics.

  c — Include condition statistics.

  e — Include expression statistics.

  f — Include finite state machine statistics.

  s — Include statement statistics.

  t — Include toggle statistics.

  To report extended toggle coverage, ensure that you have compiled (vlog/vcom) with the -code x argument, then use coverage report with -code t.

- -codeAll

  (optional) Specifies the command apply to all coverage types. Equivalent to -code bcestf.

- -nocomment | -comment

  (optional) Exclude/include comments from the report. By default, comments are included.

- -[no]concurrent | -[no]immediate

  (optional) Only concurrent assertions are selected when the -concurrent switch is used. Only immediate assertions are selected when the -immediate switch is used. These are mutually exclusive options. If neither switch is specified, both concurrent and immediate assertions are selected. Applying -no to either switch reverses the effect.

- -config

  (optional) Specifies that the current configuration of each cover directive be included in the report.

- -covered

  (optional) Displays covered bins only. This argument is used only for covergroups in coverage reports and the GUI. If any covergroup item survives the covergroup filters applied, its ancestors are displayed in the report and the GUI.

- -coverenhanced

  (optional) Enables non-critical functionality which might change the appearance or content of coverage metrics. This argument only has an effect in letter releases (10.0a, 10.0b, and so on). In major releases (10.0, 10.1, and so on), all coverage enhancements present in previous letter release streams are enabled by default, and -coverenhanced is no longer necessary to enable these enhancements. Bug fixes important to the correctness of coverage numbers are always enabled by default, with no need for -coverenhanced. Since the exact nature of -coverenhanced varies from release to release, the details of the enhancements it enables are present in the product release notes rather than in the Command Reference. For these details, search the release notes using the string "coverenhanced".

- -cvg

  (optional) Reports only covergroup coverage data.

- -cvp

  (optional) Reports only specified coverpoint or cross.

- -details [-metricanalysis]

  (optional) Includes details associated with each coverage item in the output (FEC). By default, details are not provided.

  -metricanalysis — when UDP coverage is enabled, prints sum-of-product and basic sub-condition heuristic metrics from UDP expression/condition view. It reports hit counts for all rows in UPD table. To improve coverage numbers, find rows with 0 hits and exercise the inputs accordingly. Refer to "Condition and Expression Coverage" for more information on metrics.

- -directive

  (optional) Reports only cover directive coverage data.

- -du <du_name>

  (optional) Reports coverage statistics for the specified design unit. <du_name> is <library name>.<primary>(<secondary>), where the library name is optional, and secondary name is required only for VHDL. If there are parameterized instances, all are considered to match the specified design unit. Multiple -du specifications are allowed per coverage report command.

- -excluded [-pragma | -user] [-nocomment]

  (optional) Includes details on the exclusions in the specified coverage database input file. The output is structured in Tcl command format (DO file).

  By default, this option includes both user exclusions and source code pragma exclusions, unless you specify -user or -pragma.

  -pragma — When used with the -excluded argument, writes out only lines currently being excluded by pragmas. Optional.

  -user — (optional) When used with the -excluded argument, writes out files and lines currently being excluded by the coverage exclude command.

-nocomment — Removes comments from the output, which by default appear in the exclusion report created with -excluded. Comments can be added, modified and removed from the GUI.

- -failed

  (optional) Reports only failed assertions/cover directives.

- -file <filename>

  (optional) Specifies a file name for the report. Default is to write the report to the Transcript window. Environment variables may be used in the pathname.

- -filter <pattern>

  (optional) Specifies that the command affects only those objects whose name matches the specified pattern. Must be specified with the -assert or -directive argument.

  <pattern> — Specifies the character(s) to be matched in the search. Wildcards are permitted.

- -flat [-primarykey <type | path | value> [-secondarykey <type | path | value>]]

  (optional) Presents coverage items in a text report in a flat view, sorted first by the values specified with -primarykey, and second by those specified by -secondarykey. Only assertions, cover directives, covergroups, toggles and FSMs appear in the flat view of the report. By default, the type is the primary sort, and value is secondary.

- -hidecvginsts

  (optional) Removes all covergroup instances from the report.

- -hidecvginstspi0

  (optional) Removes only the covergroup instances which have per_instance set to 0 from the report.

- -html <input_ucdb> [-code [bcesf[t|x]] [-assert] [-binrhs] [-cvg] [-directive]
  [-details[=abcdefgst]] [-du <path/du>] [-hidecvginsts | -hidecvginstspi0]
  [-htmldir <outdir>] [-instance <path/instance>]
  [-nodu] [-nofec] [-noudp] [-noframes] [-noinstance]
  [-notestplan][-notimestamps] [-nozeroweights]
  [-showexcluded] [-source] [-stmtaltflow] [-summary]
  [-testhitdata] [-testlist] [-threshL <val>] [-threshH <val>] [-usecnpm] [-verbose]

  (optional) Generates an HTML coverage report on coverage data from a given UCDB file. You can use the -verbose option with -html to enable logging output for each file generated.

  The -html arguments listed below are not compatible with any other vcover report arguments, with the exception of -cvg, -assert, -directive, -binrhs, and -code.

  For details on these arguments, refer to the coverage report argument list.

  The default output filename is *index.html* in the default directory, *covhtmlreport*.

  <input_ucdb> — (optional) Specifies input UCDB file. Only one input UCDB is allowed.

-details[=abcdefgst] — (optional) Includes coverage detail pages, which are not included by default. Optionally, you can selectively include only assertions, branch, condition, cover directives, expressions, FSM, covergroups, statements and toggles, or any combination of these.

-du <path/du> — (optional) Includes only information for the specified design unit in the HTML report; by default, all design unit information is included. May specify more than one instance by specifying the -du option multiple times in the command, or using wildcards (*) or single character substitutions (?).

-hidecvginsts | -hidecvginstspi0 — (optional) Removes from the HTML report all covergroup instances, or only those having per_instance set to 0.

-htmldir <outdir> — (optional) Specifies the name of output directory for resulting UCDB (default: "covhtmlreport"). Whether you specify an output directory or the default is used, any file or directory of that name is completely removed prior to report generation to prevent possible stale data.

-instance <path/instance> — (optional) Includes only the specified instance in the HTML report; by default, all instances are included. The instance(s) appears in the report recursively, including any downward hierarchy. May specify more than one instance by specifying the -instance option multiple times in the command, or using wildcards (*) or single character substitutions (?).

-nodu — (optional) Prevents all design unit pages from being generated for the HTML report. By default, the design unit information is included in the report.

-nofec / -noudp — (optional) Prevents FEC (-nofec) and UDP (-noudp) condition and expression types from appearing in the HTML report.

-noframes — (optional) Avoids generation of JavaScript-based tree for designs with a large number of design scopes. The report comes up as a single frame containing the top-level summary page and an HTML-only design scope index page is available as a link from the top-level page.

-noinstance — (optional) Prevents all instance pages from appearing in the HTML report. By default, the instance information is included in the report.

-notestplan — (optional) Disables generation of testplan HTML pages from merged testplan UCDB files, saving time and disk space during report generation; by default, testplan information is present.

-notimestamps — (optional) Prevents timestamp information from appearing in the report.

-nozeroweights — (optional) Prevents any items with zero weights from appearing in the report, including all zero-weighted coverage items and testplan sections. Has no effect on coverage numbers.

-precision <int> — (optional) Sets the precision for the values displayed in HTML report.

-showexcluded — (optional) Shows the excluded items in the HTML report; by default excluded items are not shown.

-source — (optional) Generates the annotated source. By default, source files are suppressed for HTML. This argument is required in order to access source code related data inside the generated HTML report.

-stmtaltflow — (optional) Allows statement HTML pages to be viewable using Firefox (versions prior to 3.0), Google Chrome, or when viewing via a web server. Without this option, statement pages are only viewable with Firefox version 3.0 and later, Opera, IE7 and later, Safari, and web servers (only with IE). Use of this option slightly increases size of HMTL report.

-summary — (optional) Includes only the top summary page, the testplan summary page, and the list of tests run in the generated report.

-testhitdata — (optional) Enables generation of a table containing test-bins-hit information. When this argument is used, tables are produced for a merged UCDB (unless merged explicitly with vcover merge -totals) in the HTML report which contain information regarding which bins were hit by which tests. The table includes bin numbers which are hyperlinked to the relevant test data.

-testlist — (optional) In addition to the test summary page, this argument generates a page of information for each test that was run. By default, these pages are excluded.

-threshL <%> -threshH <val> — (optional) Specifies % of coverage at which colored cells change from red to yellow.

-threshH <%> — (optional) Specifies % of coverage at which colored cells change from yellow to green.

-usecnpm — (optional) uses the value of option.cross_num_print_missing as the display limit for uncovered bins of covergroup cross scopes. By default, all the bins appear.

-verbose — (optional) Prints out the files that are generated by the HTML report generator.

- -instance <path>

  (optional) Writes out the source file summary coverage data for the specified instance. The <path> can be used to specify a dataset other than the current dataset. (Refer to "Object Name Syntax" for instructions on how to specify a dataset.) If no dataset is specified, the current dataset is used. Only one dataset name per command invocation may be used or an error will result.

- -lang sva | psl | vhdl

  (optional) Specifies assertions of a specific language (SVA, PSL, or VHDL). You may specify multiple languages by using this option multiple times. If -lang is not specified, all three languages are selected.

- -library <libname>

  (optional) Only needs to be used when you have packages of the same name in different libraries.

- -memory

  (optional) Reports a coarse-grain analysis of capacity data for the following SystemVerilog constructs:

  - Classes
  - Queues, dynamic arrays, and associative arrays (QDAS)
  - Assertion and cover directives
  - Covergroups
  - Solver (calls to randomize())

  When combined with -cvg and -details, this command reports the detailed memory usage of covergroup. These include the current persistent memory, current transient memory, peak transient memory, and peak time of the following:

  - Per covergroup type
  - Per coverpoint and cross in the type
  - Per covergroup instance (if applicable)
  - Per coverpoint and cross in the instance (if applicable).

- -noannotate

  (optional) Removes source code from the output report. Valid for code coverage only. Not applicable with -xml argument.

- -nocollect

  (optional) Prints a report of items whose coverage was not collected. Compatible only with the -file argument, and optionally, -append. Used with -file to print the output to a specified file name, and -append to append the data to the end of the file.

- -nocomment | -comment

  (optional) The -nocomment switch prevents comments from being displayed in the output report. By default, comments appear in the report.

- -nocrossbinsummary

  (optional) Removes cross bins for each coverpoint and cross from the report output.

- -nocvpbinsummary

  (optional) Removes covered bins for each coverpoint and cross from the report output.

- -nocvgbin

  (optional) Removes covergroup bins from the report output, for the purposes of limiting report's size.

- -noexcludedhits

  (optional) By default, if any excluded items are hit during simulation an “E-hit” notification is displayed in the text report. If “E-hit” is present, it overrides the display of “E” for an exclusion or “EA”, “ECOP”, and so forth, for various autoexclusions. To disable display of “E-hit” notifications, use the –noexcludedhits option. This has the effect of making all “E” and “EA” type notifications uniformly visible, along with exclusion reasons (if available).

- -nofec

  (optional) Removes FEC expression and condition coverage data from the output report.

- -noignorebins

  (optional) Removes covergroup ignore bins from the report output.

- -nomissing

  (optional) Removes the Misses column from the report output. Also removed from the detailed text and HTML reports is the line representing the number of missing bins out of the total number of bins (“missing/total bins:”) line for coverpoints and crosses.

- -noudp

  (optional) Removes UPD expression and condition coverage data from the output report.

- -nozeroweights

  (optional) Prevents any items with zero weights from appearing in the report, including all zero-weighted coverage items. Has no effect on coverage numbers. (optional) Prevents any items with zero weights from appearing in the report, including all zero-weighted coverage items and testplan sections. Has no effect on coverage numbers.

- -option

  (optional) Includes all covergroup option and type_option values in the report. Unless your covergroup has the "option.per_instance" set to true, only the type_option is included by default. Only applicable to covergroup reports created with the -details argument.

- -package <pkgname>

  (optional) Prints a report on the specified VHDL package body. Needs to be of the form *<lib>.<pkg>*. This argument is equivalent to -du.

- -portmode (input | output | inout | internal)

  (optional) Prints a report including toggles for only the specified port types.

- -precision <int>

  (optional) Sets the decimal precision for printing functional coverage information. Valid values are from 0 to 6 and default value is 1 (one).

- -recursive [-depth <n>]

  (optional) Reports on the instance specified with -instance and every included instance, recursively. Can also be used with -details and -totals but cannot be used with -zeros.

  -depth <n>

(optional) Used with the -recursive argument, it specifies the maximum recursive depth. A depth of 1 is the same as no recursion at all.

- -samples <spec>

  (optional) Filters the sample count for covergroup types. The sample count is optionally calculated using SVCovergroupSampleInfo, set in the *modelsim.ini* file. This argument filters to include covergroups whose sample count matches the criteria given <spec>. It is a filter for covergroup types only, not instances. If covergroups are found without a sample count, a warning is issued. <spec> is one of the following four mutually exclusive options:

  [number] — sample count is equal to [number]

  -le [number] — sample count is less than or equal to number

  -ge [number] — sample count is greater than or equal to number

  [lower]-[upper] — (no white space allowed) sample count is in the range lower-upper. There is only 1 -sample option allowed per invocation.

- -setdefault [byfile | byinstance | bydu]

  (optional) Sets the coverage report default mode for the current invocation of Questa SIM. Report modes are by file (default), by instance, and by design unit.

- -severity info | note | warning | error | failure | fatal

  (optional) Specifies the assertion severity level. When -severity is specified only assertions with the same or higher severity are selected. If not specified, assertions of all severities are selected.

- -showambiguity

  (optional) When used, coverage report displays both minimum and maximum counts for any conflicting toggle data in a UCDB that results from a combined merge (vcover merge command performed with -combine).

- -showexcluded

  (optional) When used, the coverage report displays the excluded functional coverage items.

- -source <filename>

  (optional) Writes a summary of statement coverage data for a specific source file. Environment variables may be used in the pathname.

- -testattr

  (optional) Displays test attributes in the report.

- -testextract <test_name_or_pattern>

  (optional) Displays test specific results in the report. Used to combine results from multiple tests. The <test_name_or_pattern> is the test or pattern to extract. Multiple -testextract arguments can be applied in same command. This argument is compatible with reports generated in plain text and XML formats only, HTML reports are not supported. When using this argument, a header line appears at the top of the report listing test name(s) used to

generate the report. Also, the word “hit” appears in place of the count number. UCDB files store only the aggregated coverage counts from all tests, and test-specific numbers can not be reproduced.

- -totals

  (optional) Writes out a total, recursive summary of the whole design, or of the specified instance if -instance is used. Useful for tracking changes. Without this argument, the report writes out an instance summary for each of the instances. The report prints only one summary if -totals option is used. Also, when the -totals argument is specified, the alias nodes are not counted.

  The totals summary includes a covergroup bin summary. It also includes Assertion Passes and/or Successes, Failures, and Attempts, depending on whether vsim -assertcover is used (see vsim -assertcover for more details).

- -unattemptedimmed

  (optional) Causes any unexecuted immediate assertions to be considered in the coverage calculations for Total Coverage that are displayed in the coverage report. By default, any unexecuted immediate assertions are not included in the report.

- -usecnpm

  (optional) Specifies that the value of the *modelsim.ini* variable SVCrossNumPrintMissingDefault is used in report. By default, all cross bins are displayed in the report.

- -uselimit

  (optional) Displays the maximum covergroup bin counts according to the limit specified by SV, either in the source code where covergroup is declared, or by SV’s default value. By default, the report displays actual bin counts.

- -verbose

  (optional) Prints a report listing all the integer values and their counts an integer toggle encounters during the run. List will include the number of active assertion threads (Active Count) and number of active root threads (Peak Active Count) that have occurred up to the current time.

- -xml

  (optional) Outputs report in XML format. A report created with -xml does not contain source file lines (calls -noannotate implicitly). This implicitly sets the -details argument. Refer to “Coverage Reports” for more information.

- -zeros

  (optional) Writes out a file-based summary of lines, including file names and line numbers, that have not been executed (zero hits), annotates the source code, and supports the -source and -instance options. Cannot be used in tandem with the -recursive argument.

  For covergroups, this argument applies to coverpoint and cross bins only.

For a detailed report that includes line numbers, use: coverage report -zeros -details.

Examples

- Write a coverage report with all coverage types except toggles :

```
coverage report -code bcefs
```

- Write a top-level summary of the number of files, statements, branches, hits, and signal toggles, assertions and covergroup bins to *myreport.txt*.

```
coverage report -totals -file myreport.txt
```

- Write detailed branch, condition, and statement statistics, without associated source code, to the transcript window.

```
coverage report -details -noannotate -code bcs
```

- Write a summary of code coverage for all instances to the Transcript window.

```
coverage report -byinstance
```

- Write code coverage details of all instances in the design to *myreport.txt*. The -details argument reports coverage statistics for each statement, branch, condition and expression.

```
coverage report -details -byinstance -file myreport.txt
```

- Write code coverage details of one specific instance to the Transcript window.

```
coverage report -details -instance /top/p
```

- Write toggle data from the test *clyde40ns*, listed by design unit, including both toggled and untoggled signals.

```
coverage report -details -testextract clyde40ns -bydu -code t -all
```

- Write both pragma and user-based exclusions to the transcript window as follows:

```
coverage report -excluded

# coverage report -excluded
#     src/delta/delta.vhd
#         693-696
#         711-806
#     src/delta/micro.v
#         110-124
#     src/delta/pre.v
#         216-217
#     src/delta/testdel.vhd
#         1178-1274
#     src/delta/tx.vhd
#         148-149
```

- Write both pragma and user-based exclusions to the transcript window in TCL format as follows:

```
# coverage report -excluded
# coverage exclude -add src/delta/delta.vhd 693-696 711-806
# coverage exclude -add src/delta/micro.v 110-124
# coverage exclude -add src/delta/pre.v 216-217
# coverage exclude -add src/delta/testdel.vhd 1178-1274
# coverage exclude -add src/delta/tx.vhd 148-149
```

- Write a summary of coverage by source file for coverage less than or equal to 90%.

```
coverage report -below 90 -file myreport.txt
```

- Write a list of statements with zero coverage to *myzerocov.txt*.

```
coverage report -zeros -byinstance -file myzerocov.txt
```

Related Topics

- Code Coverage
- Generating HTML Coverage Reports
- coverage save
- vcover report
- coverage analyze
- coverage attribute
- coverage ranktest
- coverage tag
- coverage goal
- coverage weight
- vcover merge
- vcover ranktest
- vsim

coverage save

The coverage save command is used to save the coverage results of the specified type from a simulation to the unified coverage database (UCDB). If no type is specified, then all types are saved into the database.

While code coverage data can also be saved with the $coverage_save_mti ($coverage_save) system task (Refer to "SystemVerilog System Tasks and Functions" in the User's Manual), the coverage save command is the preferred method of saving coverage data.

This command and its arguments can be used during live simulation or post-processing. "No Design" mode is used to save test data into a UCDB that can subsequently be loaded, using the -empty switch.

The report displays code coverage data from generate blocks.

Syntax

In Live Simulation mode:

coverage save [-instance <path>] [-code {b | c | e | f | s | t}...] [-codeAll]
[-du <du_name>] [-empty <my_test_ucdb>] [-testname <test_name>] [-norecursive]
[-onexit] [-seed] [-ovmtestname] [-uvmtestname] [-seed]
[-assert] [-coveredfcov] [-cvg] [-directive] [-help] <file_name>

In Coverage View (post-processing) mode:

coverage save [-instance <path>] [-code {b | c | e | f | s | t}...] [-codeAll]
[-empty <my_test_ucdb>] [-testname <test_name>] [-norecursive] [-onexit]
[-assert] [-coveredfcov] [-cvg] [-directive] [-help] <file_name>

In "No Design" mode:

coverage save -empty <my_test_ucdb> [-help]

Arguments

- -assert

  (optional) Save only assertion coverage data.

- -code {b | c | e | f | s | t}...

  (optional) Save only the designated coverage type: b=branch coverage; c=condition coverage; e=expression coverage; f=Finite State Machine coverage; s=statement coverage; t=toggle. More than one coverage type can be specified with a single -code argument (example: "-code bces").

- -codeAll

  (optional) Specifies the command apply to all coverage types. Equivalent to -code bcestf or no specification of type at all.

- -coveredfcov

  (optional) Save only non-zero covergroup bins and covered cover directives into the UCDB. Only supported during live simulations, not in Coverage View mode.

- -cvg

  (optional) Save only covergroup coverage data.

- -directive

  (optional) Save only cover directives coverage data.

- -du <du_name>

  (optional) Saves coverage statistics for the specified design unit. Only supported during live simulation, not in Coverage View mode.

  <du_name> is <library name>.<primary>(<secondary>), where the <library name> is optional and can be a simple leaf physical library name (without any hierarchy) or a full hierarchical physical library, and the <secondary> name is required only for VHDL. If there are parameterized instances, all are considered to match the specified design unit.

- -empty <my_test_ucdb>

  (optional) Causes only test data to be saved into the UCDB. As a standalone function, you enter "coverage save -empty <my_ucdb_name>" at a Questa TCL prompt, which produces a UCDB, with a default test record and a testname matching the specified name of the UCDB file. This file can then to loaded into viewcov mode and have other attributes added to the test record or the default ones modified, and then re-saved.

- -help

  (optional) Lists the description and syntax for the **compare save** command in the Transcript window.

- -instance <path>

  (optional) Saves coverage data for only a specified instance and any of its children, recursively. Use the -norecursive argument to exclude data from instance children. <path> is a path to the instance. You can specify more than one instance during live simulation but only one instance can be specified in Coverage View mode. <path> can also be used to specify a dataset other than the current dataset. (Refer to "Object Name Syntax" for instructions on how to specify a dataset.) If no dataset is specified, the current dataset is used. Only one dataset name per command invocation may be used or an error will result.

- -norecursive

  (optional) Saves coverage data while excluding data from children of the specified instance or design unit.

- -onexit

  (optional) Causes Questa SIM to save coverage data automatically when the simulator exits.

- -ovmtestname

  (optional) With this argument, when an OVM testname has passed to the vsim command ("vsim +OVM_TESTNAME=<myovm>"), the coverage save command uses the OVM test name as the UCDB test name, instead of the default testname (<file_name>). Only supported during live simulation, not in Coverage View mode. Not compatible with -testname. If both -ovmtestname and -uvmtestname are specified, the UVM testname is used.

  For information on setting the testname within OVM/UVM, see Saving Coverage with UVM/OVM Test Name.

- -prune <inst_path>

  (optional) Ignores hierarchies specified by <inst_path>. May appear multiple times in a coverage save command.

- -seed

  (optional) When used in conjunction with a simulation run with vsim -seed, this switch causes "_<seed_value>" to be appended to the saved testname. Only supported during live simulation, not in Coverage View mode. If no seed value has been set, zero (0) is applied. Not compatible with -testname.

- -testname <test_name>

  (optional) Saves coverage data with a specified test name (<test_name>), instead of the default test name which is derived from the specified output <file_name>. Valid characters for the <test_name> include only alphanumeric characters and the underscore character (_).

- -uvmtestname

  (optional) With this argument, when an UVM testname has passed to the vsim command ("vsim +UVM_TESTNAME=<myovm>"), the coverage save command uses the UVM test name as the UCDB test name, instead of the default testname (<file_name>). Only supported during live simulation, not in Coverage View mode. Not compatible with -testname.

  For further details on setting the testname within UVM, see Saving Coverage with UVM/OVM Test Name.

- <file_name>

  (required) Designates the name of the database file to save. Required. The command does not assign a suffix by default. If you want to save it with a .ucdb suffix, you must specify it in this argument.

Examples

- Save data from the current simulation into *myfile1.ucdb*:

  ```
  coverage save myfile1.ucdb
  ```

- Save data from the current simulation into *myfile1*, without a file suffix:

```
coverage save myfile1
```

- Save data from current simulation (into *somefile.ucdb)* when the simulator exits:

```
coverage save -onexit somefile
```

- Save data for a specific design unit or instance in the design and all its children:

```
coverage save -instance ./path/inst1 mycov
```

- Save UCDB (*mycov*) which contains *./path/inst1* hierarchy with a testname of *mytest*:

```
coverage save -instance ./path/inst1 -testname mytest mycov
```

- Save only test data to *mytest.ucdb*:

```
coverage save -empty mytest
```

Related Topics

- Code Coverage
- Coverage and Verification Management in the UCDB
- coverage analyze
- coverage attribute
- coverage report
- coverage save
- coverage tag
- vcover merge
- vcover ranktest

coverage tag

The coverage tag command can be used to create (or delete) "links" between different objects in a coverage database to specific coverage objects in a testplan. A tag is a simple string associated with the object, thus objects sharing the same tag name are linked together.

If you use the coverage tag command without the -tagname argument, it displays a list of tags, which is essentially a general purpose hierarchy browser. See examples below.

Links can be also be created from the a coverage object back to the testplan through a process called “backlinking”.

Refer to“Backlink Coverage Items to Plan with SV Attributes” for details.

This command is only available during post-simulation processing, when a UCDB file is opened with vsim -viewcov.

The coverage tag command itself defines the following:

- the portion of the UCDB that is being acted upon. You must specify just one of the following per command use:
  - the testplan (-plansection)
  - the context tree within the specified UCDB (-path)
  - the testrecords (-testrecord)
  - the design units (-duname)
- the action is to be taken:
  - display tags
  - create tags (-tagname)
  - delete tags (-delete with -tagname)
- the objects to be included in that action (filters)

Usage notes:

- You can add multiple tags on any object to mark it as a member of different categories.
- Tags can be used for primitive (non-hierarchical) grouping of coverage in the database.
- If the -cvgmatch, -cvginstmatch, -cvpmatch, and/or -crossmatch switches are used together, the lowest-level object is tagged, but the higher-level matches must be satisfied, too, before they are tagged. Thus, the matching requirements of the multiple arguments are effectively ANDed. The order of matching from lowest to highest are: -cvpmatch/-crossmatch, -cvginstmatch, -cvgmatch. See examples below.

Syntax

To create links (tags):

coverage tag <UCDB_portion> -tagname <name> [<filters>]

To remove links:

coverage tag <UCDB_portion> -delete -tagname <name> [-tagname <name>]... [<filters>]

To display links:

coverage tag [<UCDB_portion>] [<filters>]

Where <UCDB_portion> refers to the portion of the UCDB being acted upon:

(-plansection <path> | -testrecord <test_name> | -path <path> [<path>...] | -duname <lib>.<duname>)
[-recursive]

Where <filters> can be:

Tag filter:

[-tagmatch <string>]

Coverage type related filters:

[-assert] [-code {b | c | e | s | f | t }...] [-codeAll] [-cvg] [-directive]

Path filters:

([-select instance] [<pattern_filters>]) | [<bin_filters>]

where <pattern_filters> are:

([-match <string>] | [-cvgmatch <string>])
[-cvginstmatch <string>] [-crossmatch <string>] [-crossmatch <string>]

and <bin_filters> are:

-bins [-linerange [ln | ln-ln]...] [-item [bces] [<index> | <index>-<index>]...]

Arguments

- -assert

  (optional) Restricts the action of the command to assertion data.

- -bins

  (optional) By default, bins/covers are not selected for tagging. The -bins switch enables the selection of bins and/or covers. Use of the -bins argument prevents any scope/test from being selected. In other words, ONLY bins/covers will be selected when you use -bins. In order to select statement bins, you must also specify -code s.

  Paths may contain wildcards, so take care not to select more you desire. See Examples for further details.

- -code {b | c | e | s | f | t }...

  (optional) Restricts the action of the command to corresponding code coverage types: branch, condition, expression, statement, toggle, FSM. More than one coverage type can be specified with a single -code argument (example: “-code bce”).

- -codeAll

  (optional) Restricts the action of the command to all coverage types. Equivalent to -code bcestf.

- -crossmatch <string>

  (optional) Specifies a match of a cover cross name against the specified covergroup instance or covergroup type. Wildcards are allowed. See “coverage tag” description for further details.

- -cvg

  (optional) Restricts the action of the command to covergroup data.

- -cvgmatch <string>

  (optional) Specifies a match of a covergroup type against the specified pattern. This argument is implicitly recursive. See “coverage tag” description for further details.

- -cvginstmatch <string>

  (optional) Specifies a match of a covergroup instance name (<option>.<name> <value>) against the specified instance. Wildcards are allowed. See “coverage tag” description for further details.

- -cvpmatch <string>

  (optional) Specifies a match of a coverpoint against the specified covergroup instance or covergroup type. Wildcards are allowed. See “coverage tag” description for further details.

- -delete

  (optional) Removes the given tags from objects specified with the -tagname argument. If you wish to delete full or partial hierarchies of items, you must use the -recursive switch.

- -directive

  (optional) Restricts the action of the command to cover directive data.

- -duname <lib>.<duname>

  (required, unless specifying either -testrecord. -plansection, or is used with -path) Specifies the command apply to the specified library.design_unit(s). Wildcards are allowed. Mutually exclusive with -testrecord, -plansection. If used with -path, the path specified must be relative to that library and design unit.

- -item [bces] [<index> | <index>-<index>]...

  (optional) Must be used in conjunction with -linerange and -bins. Restricts the action of the command to all bins/covers that fall within the indicated range of lines and item numbers. <index> is the index number of the coverage item being specified. All ranges are inclusive.

- -linerange [ln | ln-ln]...

  (optional) Restricts the action of the command to all items that fall within the indicated range of lines without respect to item numbers. All ranges are inclusive.

- -match <string>

  (optional) Recursively matches the given pattern against the specified coverage types in the entire instance tree. If -duname is specified, it matches against the specified coverage types in the design unit. Wildcards are allowed. This switch is mutually exclusive with the following other coverage tag arguments: -recursive, coverage type filters (-assert, -codeAll, -cvg, and so forth.), -cvgmatch, -cvpmatch, and -cvginstmatch.

- -path <path> [<path>...]

  (required, unless specifying either -testrecord or -plansection, or used with -duname) Specifies a path to which the tag is applied in the UCDB. More than one path may be specified. Wildcards are allowed. You may specify "-" to indicate that the path should be ignored. Mutually exclusive with -plansection and -testrecord. If used with -duname, the path specified must be relative to that design unit.

  This argument is useful when path is specified in the Link column of a verification plan. For more information, refer to the "xml2ucdb.ini Configuration File" appendix. <path> may also be used to specify a dataset other than the current dataset. (See "Object Name Syntax" for instructions on how to specify a dataset.) If no dataset is specified, the current dataset is used. Only one dataset name per command invocation may be used or an error will result.

- -plansection <path>

  (required, unless specifying either -testrecord. -duname, or -path) Specifies a testplan path to which the tag is applied. Mutually exclusive with -path, -duname and -testrecord. Wildcards are allowed.

- -recursive

  (optional) Recursively selects objects under the given scope. Must be used with only one of the following per command invocation: -plansection <path> | -testrecord <test_name> | -path <path> [<path>...] | -duname <lib>.<duname>.

  The following command:

  ```
  coverage tag -path / -recursive
  ```

  lists all of the scopes in the context tree along with their tags. It behaves as if -path /*, -path /*/*, -path /*/*/*, and so forth. were specified for as many levels as are applicable. This switch is mutually exclusive with -match.

Tip: When writing a tag, once a scope is tagged, none of its children or descendants will be; this is because coverage numbers always include descendants, so tagging descendants adds redundant coverage to a testplan.

- -select instance

  (optional) Restricts the command to apply to design instance scopes (VHDL architectures, interface instances, and so forth.).

- -tagmatch <string>

  (optional) Searches for an exact match to strings containing the tagname specified. Useful for determining all scopes (within the portion of the UCDB being acted upon) that share a particular tagname. Use of the “*” wildcard is allowed.

- -tagname <name>

  (optional) Specifies the name of the tag to which the action is applied. May be specified multiple times or not at all. When not specified, a list of all tags is printed. If used for creation or deletion of tag(s) with -recursive, a coverage type filter is required (-cvg, -assert, and so forth.).

- -testrecord <test_name>

  (required, unless specifying either -testrecord. -plansection, or -path) Specifies the name of the specific test record in the UCDB to be linked. This tag creates a link between an item in the testplan and the test record (name of individual test run) in the UCDB. Wildcards are allowed. Can only be used in conjunction with -tagname and -delete; mutually exclusive with all other arguments.

Examples

- Display all scopes, recursively, in the entire design:

  ```
  coverage tag -path / -r
  ```

  Returns information such as the following:

  ```
  # /coverpkg/statecover [covergroup]: "2.2"
  # /coverpkg/statecover/cvpi [coverpoint] (no tags)
  # /coverpkg/statecover/cvpstate [coverpoint] (no tags)
  # /coverpkg/statecover/i_x_state [cross] (no tags)
  ```

- Display all bins, recursively, in the entire design:

  ```
  coverage tag -path / -r -bins
  ```

- Display all testplan scopes, recursively, in the entire design:

  ```
  coverage tag -plan / -r
  ```

- Display scope in the design that has been tagged with “testplan.1.2.1” tagname:

  ```
  coverage tag -path / -r -tagmatch testplan.1.2.1
  ```

- Display all bins in the design that have “plan” in the tagname:

  ```
  coverage tag -path / -r -tagmatch *plan* -bins
  ```

- Display all testplan scopes that have “plan” in the tagname:

```
coverage tag -plan / -r -tagmatch *plan*
```

- Tag a given covergroup "/a/b/cvg0" and a given instance "/a/d":

```
coverage tag -tagname T1 -path /a/b/cvg0 /a/d
```

- Tag all covergroups in the instance /a/b:

```
coverage tag -tagname T1 -cvg -path /a/b/*
```

- Tag all covergroups in the subtree rooted at /a/b:

```
coverage tag -tagname T1 -recursive -cvg -path /a/b
```

- Tag all toggles whose names start with "d" inside the given design unit:

```
coverage tag -tagname T1 -code t -duname work.du3 -path d*
```

- Tag all objects matching the pattern in all design units

```
coverage tag -tagname T1 -match a* -du *
```

- Tag all branches in the given design unit; this implicitly recurses all paths in the design unit because the filter is given and no path with the -duname:

```
coverage tag -tagname T1 -code b -duname work.duname
```

- Tag all branches in any test data records that start with the pattern “sample”:

```
coverage tag -tagname T1 -testrecord sample*
```

- Tag all covergroups whose type names start with the pattern "xactor":

```
coverage tag -tagname T1 -cvgmatch xactor*
```

- Almost the same, but will tag coverpoint, cross, and covergroup instance scopes matching the pattern:

```
coverage tag -tagname T1 -cvg -match xactor*
```

- Tag a coverpoint named "cvp1" that belongs to coverage instance "cvginst1" in covergroup "cvg1"; note this will not tag coverpoints named "cvp1" that happen to exist in other contexts. Wildcard patterns are accepted, too, in which case any coverpoint-coverinstance-covergroup combination that matches are tagged as well:

```
coverage tag -tagname T1 -cvgmatch cvg1 -cvginstmatch cvginst1
   -cvpmatch cvp1
```

- Tag a coverpoint or cross matching the given pattern inside a covergroup type named "cvgtype":

```
coverage tag -tagname T1 -cvgmatch cvgtype -cvpmatch *myvar*
```

- Display all of the bins in a specified design tree:

```
coverage tag -bins -path / -recursive
```

- Display all of the scopes under 'foo_scope':

```
coverage tag -path /top/foo_scope/*
```

- Display the bin 'cover_1' under scope 'foo_scope':

```
coverage tag -bins -path /top/foo_scope/cover_1
```

- Display all bins having the form 'cover_*' under scope 'foo_scope':

```
coverage tag -bins -path /top/foo_scope/cover_*
```

- Display all of the bins under 'foo_scope':

```
coverage tag -bins -path /top/foo_scope/*
```

- Display all covers associated with the lines 200 through 500:

```
coverage tag -bins -recursive -path * -linerange 200-500
```

Related Topics

- Code Coverage
- Verification Management Browser Window
- Verification Tracker Window
- xml2ucdb.ini Configuration File appendix
- Links Between the Plan Section and the Coverage Item
- coverage analyze
- coverage attribute
- coverage exclude
- coverage ranktest
- coverage goal
- coverage report
- coverage save
- coverage unlinked
- vcover merge
- vcover ranktest

coverage testnames

The coverage testnames command displays the testnames in the UCDB file currently loaded into memory. If a merged file, it gives you a list of tests in the merged file.

This command is most useful if you use the -testextract of coverage analyze or coverage report, because it requires the test name. By default, the testname is the name of the UCDB file, though you can set it to whatever you would like. Set the test name, before saving the UCDB file, using the command "coverage attribute -test mytestname".

This command is only available during post-simulation processing, when a UCDB file is opened with vsim -viewcov.

Syntax

coverage testnames [-tcl]

Arguments

- -tcl

  (optional) Print attribute information in a tcl format.

Related Topics

- Code Coverage
- Verification Management Browser Window
- Verification Tracker Window
- coverage attribute
- coverage exclude
- coverage goal
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage analyze
- coverage weight
- vcover merge
- vcover ranktest
- vcover stats
- vcover testnames

coverage unlinked

The coverage unlinked command is used as a diagnostic tool to report on a number of items.

- testplan item with no coverage associated
- directed tests within a testplan with no coverage associated
- design coverage that has not been associated with an item in a testplan

It reports on objects that are "unlinked" for the purposes of test traceability.

This command is only available during post-simulation processing, when a UCDB file is opened with vsim -viewcov.

Tip: When applying this command to design paths, it is a good idea to use -select and other filter arguments to restrict the command to types of objects that must be linked (only covergroups and cover directives, for example). Otherwise, the command can report as "unlinked" many irrelevant items.

Syntax

coverage unlinked
{-plansection <path> | -path <path> | -du <duname> }
[-testname <test_name>] [-file <filename>] [-append] [-nosection] [-recursive]
[<coverage_type>] [<filters>]

Where <coverage type> =
[-assert] [-code {b | c | e | f | s | t}...] [-codeAll] [-cvg] [-directive]

Where <filters> =
[-select instance]
[-select name {-eq|-ne|-[n]regexp} {<string>}]
[-select weight {-lt|-gt|-le|-ge|-eq|-ne} <int>]
[-anyselected | -allselected]
[-hideexcluded] [-nozeroweights]

Arguments

- -anyselected | -allselected

  (optional: Only has an effect with multiple -select options (does not work with -select instance).) These switches are mutually exclusive and only one may be given on each command line. Use any select criteria (logical OR) or all select criteria (logical AND) to determine what is displayed in the report. Default is -allselected.

- -append

  (optional) Appends to the output file, if -file option is used.

- -assert

  (optional) Specifies the command apply to assertion data.

- -code {b | c | e | f | s | t}...

  (optional) Specifies this command apply to corresponding code coverage types: branch, condition, expression, statement, toggle, FSM. More than one code coverage type can be specified with each -code argument (example: “-code bces”).

- -codeAll

  (optional) Specifies the command apply to all coverage types. Equivalent to -code bcestf.

- -cvg

  (optional) Specifies the command apply to covergroup data.

- -directive

  (optional) Specifies the command apply to directive data.

- -du <duname>

  (optional) Restricts the report to design units matching the given specification, <duname>. Wildcards are allowed.

- -file <filename>

  (optional) Write to output file.

- -hideexcluded

  (optional) Prevents any excluded coverage items from appearing in the report.

- -nozeroweights

  (optional) Prevents any items with zero weights from appearing in the report, including all zero-weighted coverage items and testplan sections. Has no effect on coverage numbers.

- -nosection

  (optional) Prevents the display of testplan section numbers in the output.

- -path <path>

  (optional) Restricts unlinked results to design (non-testplan) paths matching the specified path. This argument can be combined with -du to be du-relative. <path> may also be used to specify a dataset other than the current dataset. (See “Object Name Syntax” for instructions on how to specify a dataset.) If no dataset is specified, the current dataset is used. Only one dataset name per command invocation may be used or an error will result.

- -plansection <path>

  (optional) Restricts the command to the given testplan path. By default, the query treats all testplan scopes in the database, as well as all linked scopes associate with the testplan scopes.

- -recursive

  (optional) Specifies that the command is applied recursively. The default is for the query to be restricted to the single object or objects specified in the command.

- -select instance

  (optional) Includes design instances only in the unlinked items report: Verilog module, program, and interface instances, VHDL architectures, and package instances.

- -select name {-eq|-ne|-[n]regexp} {<string>}

  (optional) Displays all unlinked coverage and/or testplan items, as specified with either -plan, -path or -du arguments. The -select name filter must be used with -eq, -ne, or -[n]regexp arguments and a string, and <string> can be a pattern with wildcards.

  - -eq is "equal to"
  - -ne is "not equal to"
  - -regexp is "regular expression match", and -nregexp is "regular expression does not match"

    The -regexp and -nregexp options use Tcl regular expression matching.

- -select weight {-lt|-gt|-le|-ge|-eq|-ne} <int>

  Filters the command so that it finds unlinked item(s) whose weight(s) are less than, greater than, match, and so forth, the specified weight <int>.

- -testname <test_name>

  (optional) Finds unlinked tests within a testplan. Wildcards are allowed. Use this argument to ensure that a directed test ran within a testplan.

Examples

- Find testplan sections which are unlinked — testplans that do not share a tag with a non-testplan scope or vice-versa:

```
coverage unlinked -plansection / -r
```

- Find any unlinked covergroup or cover directive:

```
coverage unlinked -path / -cvg -dir -r
```

- Find design units that are unlinked:

```
coverage unlinked -du *
```

- Find everything under an instance tree that is unlinked, does not include design units:

```
coverage unlinked -path / -r
```

Related Topics

- Code Coverage
- Verification Management Overview
- Verification Management Browser Window
- Verification Tracker Window
- coverage analyze
- coverage attribute
- coverage exclude
- coverage goal
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage testnames
- coverage weight
- vcover merge
- vcover ranktest
- vcover stats

coverage weight

The coverage weight command is used to:

- Set or display global per-type weights
- Set or display scope weights

Setting either the per-type or scope weights has an effect on total coverage calculations.

Weights are displayed by entering a coverage weight command without an <integer_weight>.

Specifically, the command sets the overall weight for covergroups (by instance, or by design unit) and weights for individual items (design units, instances, assertions, and/or cover directives, and so forth).

Use the -plansection, -path, and -du arguments to set the weights for individual coverage items, covergroups, testplan items, design instances, or design units.

Setting weights for individual items affects coverage the same as option.weight or type_option.weight.

Total coverage calculations are exposed with the coverage analyze -total command; this includes total coverage numbers for testplans in the Test Tracking tab in the "Verification Management Browser Window" or "Verification Tracker Window".

For more information regarding type coverage statistics gathered, refer to "Calculation of Total Coverage".

Syntax

Setting global per-type weights

coverage weight [-bydu] [-byinstance] [-type]
[-attempted] [-active] [-disabled] [-fail] [-pass] [-vpass]
[-code {b | c | e | f | s | t}...] [-codeAll] [-assert] [-cvg] [-directive] [-fstate] [-ftrans]
<integer_weight>

Setting weight for individual items, either in the design or testplan hierarchy

coverage weight {-du <du_name> | -path <path> | -plansection <section_name>}
<integer_weight>

To display currently set weights, specify any coverage weight command without **<integer_weight>**.

Arguments

- -assert

  (optional) Sets weight for all assertion coverage sub-types.

- Assertion directive sub-types arguments are:

  -attempted

(optional) Sets weight for ‘attempted’ assertion directives, per instance.

-active

(optional) Sets weight for ‘active’ assertion directives, per instance.

-disabled

(optional) Sets weight for ‘disabled’ assertion directive, per instance.

-fail

(optional) Sets weight for assertion directive failures, per instance.

-pass

(optional) Sets weight for assertion directive passes, per instance.

-vpass

(optional) Sets weight for assertion directive vacuous passes, per instance.

Note

Only the use of -fail and -pass have an effect in the simulator, because all other assertion sub-types do not contribute to coverage calculations. However, all types are stored in the database.

- -bydu

  (optional) Modifier used to set per-du (code coverage only)

- -byinstance

  (optional) Modifier used to set a per-instance goal (code coverage and covergroup).

- -code {b | c | e | f | s | t}...

  (optional) Sets weight for code coverage data for coverage type: b=branch coverage; c=condition coverage; e=expression coverage; s=statement coverage; t=toggle; f=Finite State Machine coverage. More than one coverage type can be specified in a single -code argument (example: “-code bces”).

- -codeAll

  (optional) Sets weight for all code coverage types. Equivalent to -code bcestf.

- -cvg

  (optional) Sets weight for covergroup coverage type, summarized both by instance and by type.

- -directive

  (optional) Sets weight for cover directive coverage type only.

- -du <du_name>

  (optional) Sets weight for a given design unit. Mutually exclusive with -path and -plansection. Cannot be combined with any other arguments besides <integer_weight>.

- -fstate

  (optional) Sets weight for FSM state coverage.

- -ftrans

  (optional) Sets weight for FSM transition coverage.

- <integer_weight>

  (required in order to set weights; if left unspecified, command prints weights) Specifies the value for the weight: must be a natural integer, greater than or equal to 0.

  A weight of 0 turns off the coverage summary for the specified item or covergroup.

- -path <path>

  (optional) Sets the weight for a given coverage/design object. Mutually exclusive with -du and -plansection. Cannot be combined with any other arguments besides <integer_weight>. <path> may also be used to specify a dataset other than the current dataset. If no dataset is specified, the current dataset is used. Only one dataset name per command invocation may be used or an error will result.

- -plansection <section_name>

  (optional) Sets the weight for a specified testplan section. Mutually exclusive with -du and -path. Cannot be combined with any other arguments besides <integer_weight>.

- -type

  (optional) Sets weight for covergroup type coverage. The -type argument summarizes by type only, whereas the -cvg argument summarizes both by type and by instance.

Related Topics

- Code Coverage
- Coverage and Verification Management in the UCDB
- Verification Management Browser Window
- Verification Tracker Window
- coverage analyze
- coverage attribute
- coverage exclude
- coverage goal
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage testnames
- coverage unlinked

dataset alias

This command maps an alternate name (alias) to an open dataset. A dataset can have any number of aliases, but all dataset names and aliases must be unique even when more than one dataset is open. Aliases are not saved to the *.wlf* file and must be remapped if the dataset is closed and then re-opened.

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

dataset alias **<dataset_name>** [<alias_name>]

Arguments

- <dataset_name>

  (required) Specifies a dataset name or currently assigned dataset alias. Must be specified as the first argument to the dataset alias command. Returns a list of all aliases mapped to the specified dataset file when specified without <alias_name>.

- <alias_name>

  (optional) Specifies string to assign to the dataset as an alias. Wildcard characters are permitted.

Examples

Assign the alias name “bar” to the dataset named “gold.”

```
dataset alias gold bar
```

Related Topics

- dataset clear
- dataset close
- dataset config
- dataset info
- dataset list
- dataset open
- dataset rename
- dataset restart
- dataset save
- dataset snapshot

dataset clear

All event data is removed from the current simulation WLF file, while retaining all currently logged signals. Subsequent run commands will continue to accumulate data in the WLF file.

This command applies only to WLF based simulation datasets.

This command has no effect on coverage (UCDB) datasets.

If the command is executed when no design is loaded then the error: “Dataset not found:sim” is returned. If the command is executed when a design is loaded, then the “sim:” dataset is cleared, irrespective of which dataset is currently set. Clearing the dataset will clear any open wave window based on the “sim:”.

Syntax

dataset clear

Arguments

None

Examples

Clear data in the WLF file from time 0ns to 100000ns, then log data into the WLF file from time 100000ns to 200000ns.

```
add wave *
run 100000ns
dataset clear
run 100000ns
```

Related Topics

- dataset alias
- dataset close
- dataset config
- dataset info
- dataset list
- dataset open
- dataset rename
- dataset restart
- dataset save
- dataset snapshot
- log
- Recording Simulation Results With Datasets

dataset close

This command closes an active dataset. To open a dataset, use the dataset open command.

Syntax

dataset close {<dataset_name> | -all}

Arguments

- <dataset_name> | -all

  (required) Closes active dataset(s).

  <dataset_name> — Specifies the name of the dataset or alias you wish to close.

  -all — Closes all open datasets and the simulation.

Related Topics

- dataset alias
- dataset clear
- dataset config
- dataset info
- dataset list
- dataset open
- dataset rename
- dataset restart
- dataset save
- dataset snapshot

dataset config

This command configures WLF parameters for an open dataset and all aliases mapped to that dataset.

This command has no effect on coverage datasets (UCDB).

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

dataset config <dataset_name> [-wlfcachesize [<n>]] [-wlfdeleteonquit [0 | 1]] [-wlfopt [0 | 1]]

Arguments

- <dataset_name>

  (required) Specifies a open dataset or dataset alias you wish to configure. Must be specified as the first argument to the dataset config command.

- -wlfcachesize [<n>]

  (optional) Sets the size, in megabytes, of the WLF reader cache. Does not affect the WLF write cache.

  <n> — Any non-negative integer, in MB where the default is 256.

  If you do not specify a value for <n>, this switch returns the size, in megabytes, of the WLF reader cache.

- -wlfdeleteonquit [0 | 1]

  (optional) Deletes the WLF file automatically when the simulation exits. Valid for the current simulation dataset only.

  0 — Disabled (default)

  1 — Enabled

  If you do not specify an argument, this switch returns the current setting for the switch.

- -wlfopt [0 | 1]

  (optional) Optimizes the display of waveforms in the Wave window.

  0 — Disabled

  1 — Enabled (default)

  If you do not specify an argument, this switch returns the current setting for the switch.

Examples

Set the size of the WLF reader cache for the dataset “gold” to 512 MB.

```
dataset config gold -wlfcachesize 512
```

Related Topics

- dataset alias
- dataset clear
- dataset close
- dataset config
- dataset info
- dataset list
- dataset open
- dataset rename
- dataset restart
- dataset save
- dataset snapshot
- WLF File Parameter Overview
- vsim

dataset current

This command activates the specified dataset and sets the GUI context to the last selected context of the specified dataset. All context dependent GUI data is updated and all context dependent CLI commands start working with respect to the new context.

Syntax

dataset current [<dataset_name>]

Arguments

- <dataset_name>

  (optional) Specifies the dataset name or dataset alias you want to activate. If no dataset name or alias is specified, the command returns the name of the currently active dataset.

Related Topics

- dataset alias
- dataset clear
- dataset close
- dataset info
- dataset list
- dataset open
- dataset rename
- dataset restart
- dataset save
- dataset snapshot
- WLF File Parameter Overview
- vsim

dataset info

This command reports a variety of information about a dataset.

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

dataset info {name | file | exists} <dataset_name>

Arguments

- {name | file | exists}

  (required) Identifies what type of information you want reported.

  Only one option per command is allowed. The current options include:

  name — Returns the name of the dataset. Useful for identifying the real dataset name of an alias.

  file — Returns the name of the file associated with the dataset.

  exists — Returns "1" if the dataset is currently open, "0" if it does not.

  Must be specified as the first argument to the dataset info command.

- <dataset_name>

  (optional) Specifies the name of the dataset or alias for which you want information. If you do not specify a dataset name, Questa SIM uses the dataset of the current environment.

Related Topics

- dataset alias
- dataset clear
- dataset close
- dataset config
- dataset list
- dataset open
- dataset rename
- dataset restart
- dataset save
- dataset snapshot
- environment

dataset list

This command lists all active datasets.

Syntax

dataset list [-long]

Arguments

- -long

  (optional) Lists the dataset name followed by the *.wlf* file to which the dataset name is mapped.

Related Topics

- dataset alias
- dataset clear
- dataset close
- dataset config
- dataset info
- dataset open
- dataset rename
- dataset restart
- dataset save
- dataset snapshot

dataset open

This command opens a WLF file (either the currently running *vsim.wlf* or a saved WLF file) and/or UCDB file (representing coverage data) and assigns it the logical name that you specify.

The file may be the WLF file for a currently running simulation. To close a dataset, use the dataset close command.

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

dataset open <file_name> [<dataset_name>]

Arguments

- <file_name>

  (required) Specifies the file to open as a view-mode dataset. Must be specified as the first argument to the dataset open command. Specify *vsim.wlf* to open the currently running WLF file.

- <dataset_name>

  (optional) Specifies a name for the open dataset. This is a name that will identify the dataset in the current session. By default the dataset prefix will be the name of the specified file.

Examples

Open the dataset file *last.wlf* and assign it the name *test*.

```
dataset open last.wlf test
```

Related Topics

- dataset alias
- dataset clear
- dataset close
- dataset config
- dataset info
- dataset list
- dataset rename
- dataset restart
- dataset save
- dataset snapshot
- vsim -view option

dataset rename

This command changes the name of a dataset to the new name you specify.

Arguments to this command are order dependent. Follow the order specified in the Syntax section.

Syntax

dataset rename **<dataset_name> <new_dataset_name>**

Arguments

- <dataset_name>

  Specifies the existing name of the dataset.

- <new_dataset_name>

  Specifies the new name for the dataset.

Examples

Rename the dataset file "test" to "test2".

```
dataset rename test test2
```

Related Topics

- dataset alias
- dataset clear
- dataset close
- dataset config
- dataset info
- dataset list
- dataset open
- dataset restart
- dataset save
- dataset snapshot

dataset restart

This command unloads the specified dataset or currently active dataset and reloads the dataset using the same dataset name.

The contents of Wave and other coverage windows are restored for UCDB datasets after a reload.

Syntax

dataset restart [<file_name>]

Arguments

- <file_name>

  (optional) Specifies the file to open as a dataset. If <filename> is not specified, the currently active dataset is restarted.

Related Topics

- dataset alias
- dataset clear
- dataset close
- dataset config
- dataset info
- dataset list
- dataset open
- dataset rename
- dataset save
- dataset snapshot

dataset save

This command writes data from the current simulation to the specified file. This lets you save simulation data while the simulation is still in progress.

Arguments to this command are order dependent. Follow the order specified in the Syntax section.

This command is equivalent to the coverage save command for coverage datasets.

Syntax

dataset save **<dataset_name> <file_name>**

Arguments

- <dataset_name>

  (required) Specifies the name of the dataset you want to save.

- <file_name>

  (required) Specifies the name of the file to save.

Examples

Save all current log data in the sim dataset to the file *gold.wlf*.

```
dataset save sim gold.wlf
```

Related Topics

- dataset alias
- dataset clear
- dataset close
- dataset config
- dataset info
- dataset list
- dataset open
- dataset rename
- dataset restart
- dataset snapshot

dataset snapshot

This command saves data from the current WLF file (*vsim.wlf* by default) at a specified interval. It provides you with sequential or cumulative "snapshots" of your simulation data.

This command does not apply to coverage datasets (UCDB).

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

dataset snapshot [-dir <directory>] [-disable] [**-enable**] [-file <file_name>] [-filemode {overwrite | increment}] [-mode {cumulative | sequential}] [-report] [-reset] {**-size <file_size>** | **-time <n> [<unit>]**}

Arguments

- -dir <directory>

  (optional) Specifies a directory into which the files should be saved. Either absolute or relative paths may be used. Default is to save to the current working directory.

- -disable

  (optional) Turns snapshotting off. All dataset snapshot settings from the current simulation are stored in memory. All other options are ignored after you specify -disable.

- -enable

  (optional) Turns snapshotting on. Restores dataset snapshot settings from memory or from a saved dataset. (default)

- -file <file_name>

  (optional) Specifies the name of the file to save snapshot data.

  <file_name> — A specified file name where the default is *vsim_snapshot.wlf*. The suffix *.wlf* will be appended to specified filename and, possibly, an incrementing suffix.

  When the duration of the simulation run is not a multiple of the interval specified by -size or -time, the incomplete portion is saved in the file *vsim.wlf*.

- -filemode {overwrite | increment}

  (optional) Specifies whether to overwrite the snapshot file each time a snapshot occurs.

  overwrite — (default)

  increment — A new file is created for each snapshot. An incrementing suffix (1 to n) is added to each new file (for example, *vsim_snapshot_1.wlf*).

- -mode {cumulative | sequential}

  (optional) Specifies whether to keep all data from the time signals are first logged.

  cumulative — (default)

sequential — The current WLF file is cleared every time a snapshot is taken.

- -report

  (optional) Lists current snapshot settings in the Transcript window. All other options are ignored if you specify -report.

- -reset

  (optional) Resets values back to defaults. The behavior is to reset to the default, then apply the remainder of the arguments on the command line. See examples below. If specified by itself without any other arguments, -reset disables dataset snapshot and resets the values.

- -size <file_size>

  (Required if -time is not specified.) Specifies that a snapshot occurs based on WLF file size. Must be specified as the final argument to the dataset snapshot command.

  <file_size> — Size of WLF file in MB.

- -time <n> [<unit>]

  (Required if -size is not specified.) Specifies that a snapshot occurs based on simulation time. Must be specified as the final argument to the dataset snapshot command.

  <n> — Any positive integer.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <limit> and <unit> within braces ({}).

Examples

- Create the file *vsim_snapshot_<n>.wlf* that is written to every time the current WLF file reaches a multiple of 10 MB (i.e., at 10 MB, 20 MB, 30 MB, etc.).

  dataset snapshot -size 10

- Similar to the previous example, but in this case the current WLF file is cleared every time it reaches 10 MB.

  dataset snapshot -size 10 -mode sequential

- Assuming simulator time units are ps, this command saves a file called *gold_<n>.wlf* every 1000000 ps. If you run the simulation for 3000000 ps, three files are saved: *gold_1.wlf* with data from 0 to 1000000 ps, *gold_2.wlf* with data from 1000000 to 2000000, and *gold_3.wlf* with data from 2000000 to 3000000.

  dataset snapshot -time 1000000 -file gold.wlf -mode sequential -filemode increment

  Because this example sets the time interval to 1000000 ps, if you run the simulation for 3500000 ps, a file containing the data from 3000000 to 3500000 ps is saved as *vsim.wlf* (default).

- Enable snapshotting with time=10000 and default mode (cumulative) and default filemode (overwrite).

```
dataset snapshot -reset -time 10000
```

Related Topics

- dataset alias
- dataset clear
- dataset close
- dataset config
- dataset info
- dataset list
- dataset open
- dataset rename
- dataset restart
- dataset save

delete

This command removes objects from either the List or Wave window. Arguments to this command are order dependent.

Syntax

delete list [-window <wname>] **<object_name>...**

delete wave [-window <wname>] **<object_name>...**

Arguments

- list

  Specifies the target is a list window.

- wave

  Specifies the target is a wave window.

- -window <wname>

  (optional) Specifies the name of the List or Wave window to target for the delete command. (The view command allows you to create more than one List or Wave window.) If no window is specified, the default window is used; the default window is determined by the most recent invocation of the view command and has " - Default" appended to the name.

- <object_name>...

  (required) Specifies the name of an object. Must match an object name used in an add list or add wave command. Multiple object names are specified as a space separated list. Wildcard characters are allowed. Must be specified as the final argument to the delete list and delete wave commands.

Examples

- Remove the object *vec2* from the list2 window.

  delete list -window list2 vec2

- Remove all objects beginning with the string /test from the Wave window.

  delete wave /test*

Related Topics

- add list
- add wave
- Wildcard Characters

describe

This command displays information about the following types of simulation objects and design regions in the Transcript window:

- **VHDL** — signals, variables, constants, and FILE objects.
- **Verilog** — nets and registers
- **C —** variables
- **SystemC —** signals, ports, FIFOs, and member variables of modules
- Design region

VHDL signals, Verilog nets and registers, and SystemC signals and ports can be specified as hierarchical names.

C variables can be described if you are running "C Debug", and the variables are local to the active call frame for the line in the function in the C source file where you are stopped.

For specific information related to viewing SystemC objects refer to "SystemC Object and Type Display".

Syntax

describe <name>...

Arguments

- <name>...

  (required) The name of an HDL object, SystemC signal, or C variable for which you want a description.

  Multiple object names are specified as a space separated list. Wildcard characters are allowed. HDL object names can be relative or full hierarchical names.

Examples

- Print the type of C variable *x*.

  describe x

- Print the type of what *p* points to.

  describe *p

- Print the types of the three specified signals.

  describe clk prw prdy

- Return information about /textio/INPUT.

  describe /textio/INPUT

  produces:

```
# File of
#   Unconstrained Array of
#     VHDL standard type CHARACTER
```

Related Topics

- add list
- add wave
- Wildcard Characters

disablebp

This command turns off breakpoints and when commands. To turn on breakpoints or when commands again, use the enablebp command.

Syntax

disablebp [<id#> | <label>]

Arguments

- <id#>

  (optional) Specifies the ID number of a breakpoint or when statement to disable.

  Note that C breakpoint id#s are prefixed with "c.".

- <label>

  (optional) Specifies the label name of a breakpoint or when statement to disable.

If you do not specify either of these arguments, all breakpoints and when statements are disabled.

Use the bp command with no arguments to find labels and ID numbers for all breakpoints in the current simulation. Use the when command with no arguments to find labels and ID numbers of all when statements in the current simulation.

Note

Id numbers for breakpoints and when statements are assigned from the same pool. Even if you have not specified a given id number for a breakpoint, that number may still be used for a when command.

Related Topics

- bd
- bp
- C Debug
- enablebp
- onbreak
- resume
- when

disable_menu

This command disables the specified menu within the specified window.

The disabled menu will become grayed-out and nonresponsive.

Arguments to this command are order dependent. Follow the order specified in the Syntax section.

Syntax

disable_menu **<window_name>** <menu_path>

Arguments

- <window_name>

  (required) The path of the window containing the menu. You must express the path for the Main window as "". All other window pathnames begin with a period (.) as shown in the examples below.

- <menu_path>

  (required) Name of the Tk menu-widget path. Must be specified as the final argument to the disable_menu command.

Examples

- Disable the file menu of the Main window.

  ```
  disable_menu "" File
  ```

- Disable the file menu of the mywindow window.

  ```
  disable_menu .mywindow File
  ```

Related Topics

- add_menu
- disable_menuitem
- enable_menu

disable_menuitem

This command disables a specified menu item within the specified menu path of the specified window.

The menu item will become grayed-out and nonresponsive.

Arguments to this command are order dependent. Follow the order specified in the Syntax section.

Syntax

disable_menuitem **<window_name>** **<menu_path>** <label>

Arguments

- <window_name>

  (required) Tk path of the window containing the menu.

  Note that the path for the Main window must be expressed as "". All other window pathnames begin with a period (.) as shown in the example below.

- <menu_path>

  (required) Name of the Tk menu-widget path. The path may include a submenu as shown in the example below.

- <label>

  (required) Menu item text.

Examples

- Locate the mywindow window, disable the Save Results As... menu item in the save submenu of the file menu.

```
disable_menuitem .mywindow file.save "Save Results As..."
```

Related Topics

- add_menu
- disable_menu
- enable_menu

do

This command executes the commands contained in a DO file.

A DO file can have any name and extension. An error encountered during the execution of a DO file script causes its execution to be interrupted, unless an onerror command or the OnErrorDefaultAction Tcl variable is specified with the resume command. The onbreak command is used to take action with source code breakpoint cases.

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

do <filename> [<parameter_value>...]

Arguments

- <filename>

  (required) Specifies the name of the DO file to be executed. The name can be a pathname or a relative file name. Pathnames are relative to the current working directory. Must be specified as the first argument to the do command.

  If the do command is executed from another DO file, pathnames are relative to the directory of the calling DO file. This allows groups of DO files to be stored in a separate sub-directory.

- <parameter_value>...

  (optional) Specifies values that are to be passed to the corresponding parameters $1 through $9 in the DO file. Multiple parameter values must be separated by spaces.

  If you want to make the parameters optional (for example, specify fewer parameter values than the number of parameters actually used in the file), you must use the argc simulator state variable in the DO file script. Refer to "Simulator State Variables" and "Making Script Parameters Optional".

Note

While there is no limit on the number of parameters that can be passed to macros, only nine values are visible at one time. Use the shift command to see the other parameters.

Examples

- Execute the file *macros/stimulus and* pass the parameter value 100 to $1 in the DO file.

  do macros/stimulus 100

  Where the DO file *testfile* contains the line

  ```
  bp $1 $2
  ```

  place a breakpoint in the source file named *design.vhd* at line 127.

```
do testfile design.vhd 127
```

Related Topics

- Tcl and DO Files
- Modes of Operation
- Using a Startup File
- DOPATH variable
- Saving a Transcript File as a DO File

down

This command searches for object transitions or values in the specified List window.

It executes the search on objects currently selected in the window, starting from the point of the active cursor. The active cursor moves to the found location.

Use this command to move to consecutive transitions or to find the time at which an object takes on a particular value, or an expression of multiple objects evaluates to true. See the up command for related functionality.

The procedure for using down includes three steps:

1. Click on the desired object.
2. Click on the desired starting location.
3. Issue the down command. (The seetime command can initially position the cursor from the command line, if desired.)

Returns: <number_found> <new_time> <new_delta>

Syntax

down [-expr <expression>] [-falling] [<n>] [-noglitch] [-rising] [-value <sig_value>] [-window]

Arguments

- -expr <expression>

  (optional) The List window is searched until the expression evaluates to a boolean true condition.

  <expression> — An expression that involves one or more objects, but limited to objects that have been logged in the referenced List window. An object may be specified either by its full path or by the shortcut label displayed in the List window.

  See GUI_expression_format for the format of the expression. The expression must be placed within braces.

- -falling

  (optional) Searches for a falling edge on the specified object if that object is a scalar object. If it is not a scalar object, the option will be ignored.

- <n>

  (optional) Specifies to find the nth match. If less than n are found, the number found is returned with a warning message, and the marker is positioned at the last match.

- -noglitch

  (optional) Specifies that delta-width glitches are to be ignored.

- -rising

  (optional) Searches for a rising edge on the specified object if that object is a scalar object. If it is not a scalar object, the option will be ignored.

- -value <sig_value>

  (optional) Specifies the value of the object to match.

  <sig_value> — A value specified in the same radix that the selected object is displayed. Case is ignored, but otherwise the value must be an exact string match. We do not support don't-care bits.

- -window

  (optional) Specifies an instance of the List window that is not the default. When <wname> is not specified, the default List window is used. Use the view command to change the default window.

  <wname> — The name of a List window not currently the default.

Examples

- Find the next time at which the selected vector transitions to FF23, ignoring glitches.

  ```
  down -noglitch -value FF23
  ```

- Go to the next transition on the selected object.

  ```
  down
  ```

The following examples illustrate search expressions that use a variety of object attributes, paths, array constants, and time variables. Such expressions follow the GUI_expression_format.

- Search down for an expression that evaluates to a boolean 1 when object *clk* just changed from low to high and object *mystate* is enumeration reading and object */top/u3/addr* is equal to the specified 32-bit hex constant.

  ```
  down -expr {clk'rising && (mystate == reading) && (/top/u3/addr == 32'habcd1234)}
  ```

- Search down for an expression that evaluates to a boolean 1 when the upper 8 bits of the 32-bit object */top/u3/addr* equals hex ac.

  ```
  down -expr {(/top/u3/addr and 32'hff000000) == 32'hac000000}
  ```

- Search down for an expression that evaluates to a boolean 1 when logfile time is between 23 and 54 microseconds, clock just changed from low to high, and object *mode* is enumeration writing.

  ```
  down -expr {((NOW > 23 us) && (NOW < 54 us)) && clk'rising && (mode == writing)}
  ```

Related Topics

- GUI_expression_format
- view
- seetime
- up

drivers

This command displays the names and strength of all drivers of the specified object.

The driver list is expressed relative to the top-most design signal/net connected to the specified object. If the object is a record or array, each sub-element is displayed individually.

The output from the drivers command, which is displayed in the Transcript window as a hypertext link, allowing you to right-click to open a drop-down menu and quickly add signals to various windows. It includes a "View Declaration" item to open the source definition of the signal.

Figure 2-1. drivers Command Results in Transcript

Syntax

drivers <object_name> [-source]

Arguments

- <object_name>

  (required) Specifies the name of the signal or net whose drivers are to be shown. All signal or net types are valid. Multiple names and wildcards are accepted.

- -source

  (optional) Returns the source file name and line number for each driver of the specified signal or net. If the source location cannot be determined, the value n/a is returned for that driver.

Examples

drivers /top/dut/pkt_cnt(4)

```
# Drivers for /top/dut/pkt_cnt(4):
#    St0  : Net /top/dut/pkt_cnt[4]
#      St0 : Driver /top/dut/pkt_counter/#IMPLICIT-WIRE(cnt_out)#6
```

In some cases, the output may supply a strength value similar to 630 or 52x, which indicates an ambiguous verilog strength.

Related Topics

- readers
- Verilog Language Reference Manual Std 1365-2005 section 7.10.2 "Ambiguous strengths: sources and combinations"

dumplog64

This command dumps the contents of the specified WLF file in a readable format to stdout.

The WLF file cannot be opened for writing in a simulation when you use this command. This command cannot be used in a DO file.

Syntax

dumplog64 <filename>

Arguments

- <filename>

  (required) The name of the WLF file to be read.

echo

This command displays a specified message in the Transcript window.

Syntax

echo **[<text_string>]**

Arguments

- <text_string>

  (required) Specifies the message text to be displayed. If the text string is surrounded by quotes, blank spaces are displayed as entered. If quotes are omitted, two or more adjacent blank spaces are compressed into one space.

Examples

- If the current time is 1000 ns, this command:

  echo "The time is $now ns."

  returns the message:

  ```
  The time is       1000 ns.
  ```

- If the quotes are omitted:

  echo The time is $now ns.

  all blank spaces of two or more are compressed into one space.

  ```
  The time is $now ns."
  ```

- **echo** can also use command substitution, such as:

  echo The hex value of counter is [examine -hex counter].

  If the current value of counter is 21 (15 hex), this command returns:

  ```
  The hex value of counter is 15.
  ```

edit

This command invokes the editor specified by the EDITOR environment variable. By default, the specified filename will open in the Source window.

Syntax

edit [<filename>]

Arguments

- <filename>

  (optional) Specifies the name of the file to edit. If the <filename> argument is omitted, the editor opens the current source file. If you specify a non-existent filename, it will open a new file. Either absolute or relative paths may be used.

Related Topics

- notepad
- EDITOR environment variable

enablebp

This command turns on breakpoints and when commands that were previously disabled.

Syntax

enablebp [<id#> | <label>]

Arguments

- <id#>

  (optional) Specifies a breakpoint ID number or when statement to enable.

  Note that C breakpoint id numbers have a "c" prefix.

- <label>

  (optional) Specifies the label name of a breakpoint or when statement to enable.

If you do not specify either of these arguments, all breakpoints are enabled.

Use the bp command with no arguments to find labels and ID numbers for all breakpoints in the current simulation. Use the when command with no arguments to find labels and ID numbers of all when statements in the current simulation.

Related Topics

- bd
- bp
- C Debug
- disablebp
- onbreak
- resume
- when

enable_menu

This command enables a previously disabled menu. The menu will be changed from grayed-out to normal and will become responsive. Returns nothing.

Arguments to this command are order dependent. Follow the order specified in the Syntax section.

Syntax

enable_menu **<window_name>** <menu_path>

Arguments

- <window_name>

  (required) Tk path of the window containing the menu.

  Note that the path for the Main window must be expressed as "". All other window pathnames begin with a period (.) as shown in the example below.

- <menu_path>

  (required) Name of the Tk menu-widget path.

Examples

- Enable the previously-disabled File menu of the Main window.

  enable_menu "" File

- Enable the previously-disabled File menu of the mywindow window.

  enable_menu .mywindow File

Related Topics

- add_menu
- disable_menu

enable_menuitem

This command enables a previously disabled menu item.

The menu item changes from grayed-out to normal, and becomes responsive. Returns nothing.

Arguments to this command are order dependent. Follow the order specified in the Syntax section

Syntax

enable_menuitem **<window_name>** **<menu_path>** <label>

Arguments

- <window_name>

  (required) Tk path of the window containing the menu.

  Note that the path for the Main window must be expressed as "". All other window pathnames begin with a period (.) as shown in the example below.

- <menu_path>

  (required) Name of the Tk menu-widget path. The path may include a submenu as shown in the example below.

- <label>

  (required) Menu item text.

Examples

This command locates the mywindow window and enables the previously-disabled Save Results As... menu item in the save submenu of the file menu.

```
enable_menuitem .mywindow file.save "Save Results As..."
```

Related Topics

- add_menuitem
- disable_menuitem

encoding

This command translates between the 16-bit Unicode characters used in Tcl strings and a named encoding, such as Shift-JIS. There are four encoding commands used to work with the encoding of your character representations in the GUI.

- encoding convertfrom — Convert a string from the named encoding to Unicode.
- encoding convertto — Convert a string to the named encoding from Unicode.
- encoding names — Returns a list of all valid encoding names.
- encoding system — Changes the current system encoding to a named encoding. If a new encoding is omitted the command returns the current system encoding. The system encoding is used whenever Tcl passes strings to system calls.

Syntax

encoding convertfrom <encoding_name> <string>

encoding convertto <encoding_name> <string>

encoding names

encoding system <encoding_name>

Arguments

- string

  Specifies a string to be converted.
- encoding_name

  The name of the encoding to use.

environment

This command has two forms, environment and env. It allows you to display or change the current dataset and region/signal environment.

Syntax

environment [-dataset | -nodataset] [<pathname> | -forward | -back]

Arguments

- -dataset

  (optional) Displays the specified environment pathname *with* a dataset prefix. Dataset prefixes are displayed by default.

- -nodataset

  (optional) Displays the specified environment pathname without a dataset prefix.

- <pathname>

  (optional) Specifies a new pathname for the region/signal environment.

  If omitted the command causes the pathname of the current region/signal environment to be displayed.

- -forward

  (optional) Displays the next environment in your history of visited environments.

- -back

  (optional) Displays the previous environment in your history of visited environments.

Examples

- Display the pathname of the current region/signal environment.

  env

- Change to another dataset but retain the currently selected context.

  env test:

- Change all unlocked windows to the context "test:/top/foo".

  env test:/top/foo

- Move down two levels in the design hierarchy.

  env blk1/u2

- Move to the top level of the design hierarchy.

  env /

Related Topics

- Refer to Object Name Syntax for information on specifying pathnames.
- Refer to Setting your Context by Navigating Source Files for more information about -forward and -back.

examine

This command has two forms, examine and exa. It examines one or more objects and displays current values (or the values at a specified previous time) in the Transcript window.

It can also compute the value of an expression of one or more objects.

If the design is being optimized with vopt, some of the objects listed below may not be available for viewing. See "Preserving Object Visibility for Debugging Purposes" for more information.

If you are using C Debug, **examine** can display the value of a C variable as well.

The following objects can be examined:

- **VHDL** — signals, shared variables, process variables, constants, generics, and FILE objects
- **Verilog** — nets, registers, parameters, and variables
- **C** — variables
- **SystemC** — signals, FIFOs, ports, and member variables of modules

When stopped in C code, **examine** (with no arguments) displays the values of the local variables and arguments of the current C function. For specific information related to viewing SystemC objects refer to “SystemC Object and Type Display”.

To display a previous value, specify the desired time using the -time option.

To compute an expression, use the -expr option. The -expr and the -time options may be used together.

Virtual signals and functions may also be examined within the GUI (actual signals are examined in the kernel).

The following rules are used by the examine command to locate an HDL object:

- If the name does not include a dataset name, then the current dataset is used.
- If the name does not start with a path separator, then the current context is used.
- If the name is a path separator followed by a name that is not the name of a top-level design unit, then the first top-level design unit in the design is used.
- For a relative name containing a hierarchical path, if the first object name cannot be found in the current context, then an upward search is done up to the top of the design hierarchy to look for a matching object name.
- If no objects of the specified name can be found in the specified context, then an upward search is done to look for a matching object in any visible enclosing scope up to an instance boundary. If at least one match is found within a given context, no (more) upward searching is done; therefore, some objects that may be visible from a given context will not be found when wildcards are used if they are within a higher enclosing scope.

- The wildcards '*' and '?' can be used at any level of a name except in the dataset name and inside of a slice specification.
- A wildcard character will never match a path separator. For example, */dut/** will match */dut/siga* and */dut/clk*. However, */dut** won't match either of those.

Setting the radix and radix flag values:

If no radix is specified with the examine command, the default radix and radix flags are used.

- Set the default radix with the radix command or by editing the DefaultRadix variable in the modelsim.ini file.
- Set the default radix flags value with the radix command or by editing the DefaultRadixFlags variable in the modelsim.ini file.
- Specifying examine -<radix_type> returns the value of the object in the specified radix and default radix flags value.
- Specifying examine -radix <radix_type> returns the value of the object in the specified radix.
- Specifying examine -radix [<radix_type>][, [<radix_flag>] returns the value of the object in the specified radix and radix flags.
- Specifying examine -radix <radix_flag>[,<radix_flag>] returns the value of the object in the default radix and specified radix flags.

  For example, assume a default of hexadecimal + showbase:

```
examine d
# 16'h0009
examine -binary d
# 16'b0000000000001001
examine -radix binary d
# 0000000000001001
examine -radix binary,showbase d
# 16'b0000000000001001
examine -radix hex,enumsymbolic nxt_state
# send5
examine -radix hex,enumnumeric nxt_state
# 0000000d
```

Refer to Design Object Names for more information on specifying names.

Syntax

examine **<name>…** [-delta <delta>] [-env <path>] [-event <time>] [-handle] {[-in] [-out] [-inout] | [-ports]} [-internal] [-maxlen <integer>] [-expr <expression>] [-name] [-<radix_type>] [-radix [<radix_type>][,<radix_flag>][,...]] [-radixenumnumeric | -radixenumsymbolic] [-showbase] [-time <time>] [-value]

Arguments

- <name>…

  (required except when specifying -expr.) Specifies the name of any HDL or SystemC object.

  All object types are allowed, except those of the type file. Multiple names and wildcards are accepted. Spaces, square brackets, and extended identifiers require braces; see examples below for more details. To examine a VHDL variable you can add a process label to the name. For example, (make certain to use two underscore characters):

  exa line__36/i

- -delta <delta>

  (optional) Specifies a simulation cycle at the specified time step from which to fetch the value, where the default is to use the last delta of the time step. You must log the objects to be examined using the add list, add wave, or log command for the examine command to be able return a value for a requested delta.

  <delta> — Any non-negative integer.

- -env <path>

  (optional) Specifies a path in which to look for an object name.

  <path> — The specified path to a object.

- -event <time>

  (optional) Specifies a simulation cycle at the specified event time from which to fetch the value. The event <time> refers to the event time relative to events for all signals in the objects dataset at the specified time. You must log the objects to be examined using the add list, add wave, or log command for the examine command to be able return a value for a requested event.

- -expr <expression>

  (optional) Specifies an expression to be examined. You must log the expression using the add list, add wave, or log command for the examine command to return a value for a specified expression. The expression is evaluated at the current time simulation. If you also specify the **-time** argument, the expression will be evaluated at the specified time. It is not necessary to specify <name> when using this argument. See GUI_expression_format for the format of the expression.

  <expression> — Specifies an expression enclosed in braces ({}).

- -handle

  (optional) Returns the memory address of the specified <name>. You can use this value as a tag when analyzing the simulation. This value also appears as the title of a box in the Watch window. This option will not return any value if you are in -view mode.

- -in

  (optional) Specifies that <name> include ports of mode IN.

- -out

  (optional) Specifies that <name> include ports of mode OUT.

- -inout

  (optional) Specifies that <name> include ports of mode INOUT.

- -internal

  (optional) Specifies that <name> include internal (non-port) signals.

- -maxlen <*integer*>

  (optional) Specifies the maximum number of characters in the output of the command.

  <integer> — Any non-negative integer where 0 is unlimited.

- -ports

  (optional) Specifies that <name> include all ports. Has the same effect as specifying -in, -inout, and -out together.

- -name

  (optional) Displays object name(s) and value(s). Related switch is -value.

  The lecho command will return the output of an examine command in "pretty-print" format. For example,

  lecho [examine -name clk prw pstrb]

- -<radix_type>

  (optional) Specifies the radix type for the objects that follow in the command. Retains the current flag value for the objects that follow in the command. Valid entries (or any unique abbreviations) are: ascii, binary, decimal, fpoint, hexadecimal, octal, sfixed, symbolic, time, ufixed, unsigned, and default.

  This option overrides the global setting of the default radix (the DefaultRadix variable in the modelsim.ini file).

- -radix [<radix_type>][,<radix_flag>][,...]

  (optional) Specifies the radix and/or the radix flags to be used by the examine command. The -radix <radix_type> switch can be used in place of examine -<radix_type>.

  <radix_type> — (required unless specifying <radix_flag>) Specifies a radix and clears the radix flags for the objects that follow in the command. Valid values are: ascii, binary, decimal, fpoint, hexadecimal, octal, sfixed, symbolic, time, ufixed, unsigned, default, and user- defined radix names (refer to the radix define command).

  <radix_flag> — (optional) Sets one or more radix flags on the objects that follow in the command. Multiple flags specified as a comma separated list. Must follow -<radix_type> when the two are specified together.

  Valid radix flags:

enumeric — Displays Verilog and SystemC enums as numbers (formatted by the current radix). This overrides the default behavior of always showing enums symbolically.

enumsymbolic — Restores the default behavior of displaying Verilog and SystemC enums as symbols by reversing the action of the -radixenumnumeric option.

showbase — Displays the number of bits of the vector and the radix used, where:

binary = b
decimal = d
hexadecimal = h
ASCII = a
time = t

For example, instead of simply displaying a vector value of “31”, a value of “16’h31” may be displayed to show that the vector is 16 bits wide, with a hexadecimal radix.

This option overrides the global default settings for the radix and the radix flag (the DefaultRadix and the DefaultRadixFlags in the modelsim.ini file).

- -radixenumnumeric

  (optional) Causes Verilog and SystemC enums to be displayed as numbers (formatted by the current radix). This overrides the default behavior of always showing enums symbolically.

- -radixenumsymbolic

  (optional) Restores the default behavior of displaying Verilog and SystemC enums as symbols by reversing the action of the -radixenumnumeric option.

- -showbase

  (optional) Display the number of bits of the vector and the radix used, where:

  binary = b
  decimal = d
  hexadecimal = h
  ASCII = a
  time = t

  For example, instead of simply displaying a vector value of “31”, a value of “16'h31” may be displayed to show that the vector is 16 bits wide, with a hexadecimal radix.

- -time <time>

  (optional) Specifies the time value between 0 and $now for which to examine the objects.

  <time> — A non negative integer where the default unit is the current time unit. If the <time> field uses a unit other than the current unit, the value and unit must be placed in braces. For example, the following are equivalent for ps resolution:

  exa -time {3.6 ns} signal_a
  exa -time 3600 signal_a

If an expression is specified it will be evaluated at that time. The objects to be examined must be logged via the add list, add wave, or log command in order for the examine command to be able to return a value for a requested time.

- -value

  (default) Returns value(s) as a curly-braces separated Tcl list. Use to toggle off a previous use of -name.

Examples

- Return the value of */top/bus1*.

  examine /top/bus1

- Return the value of the subelement of *rega* that is specified by the index (16). Note that you must use braces when examining subelements.examine

  {rega[16]}

- Return information about /textio/OUTPUT

  examine /textio/OUTPUT

  returns the following:

  ```
  # {STD_OUTPUT {stdout NOTPIPE} WRITE_MODE N/A}
  ```

  The output is a Tcl list with up to four elements. There are three scenarios of results:

  - If the file has not been elaborated, the result will be the following one-element list:

    ```
    # {"NOT ELABORATED"}
    ```

  - If the file is closed, the result will be the following one-element list.

    ```
    # {"CLOSED"}
    ```

  - In all other cases, the result will be a four-element list, following the format:

    ```
    # {<file_path> { <descriptor> PIPE | NOTPIPE } <file_mode>
    <file_position>}
    ```

    where,

    - <file_path> — references either the FILE declaration or corresponding file_open() call
    - <descriptor> — one of the following:

      N — operating system file descriptor resource number.

      stdin — identifying the file as stdin.

      stdout — identifying the file as stdout.

- <file_mode> — identifying the file mode, of which the file was opened, as defined in std.standard package; either READ_MODE, WRITE_MODE, or APPEND_MODE.
- <file_position> — value in bytes. For stdin or stdout files, the value will be “N/A”.

- Return the value of the contiguous subelements of *foo* specified by the slice (i.e., 20:22). Note the use of braces.

  examine {foo[20:22]}

- Note that when specifying an object that contains an extended identifier as the last part of the name, there must be a space after the closing '\' and before the closing '}'.

  examine {/top/\My extended id\ }

- In this example, the **-expr** option specifies a signal path and user-defined Tcl variable. The expression will be evaluated at 3450us.

  examine -time {3450 us} -expr {/top/bus and $bit_mask}

- Using the ${fifo} syntax limits the variable to the simple name fifo, instead of interpreting the parenthesis as part of the variable. Quotes are needed when spaces are involved; and by using quotes (“ “) instead of braces, the Tcl interpreter will expand variables before calling the command.

  examine -time $t -name $fifo "${fifo}(1 to 3)" ${fifo}(1)

- Because **-time** is not specified, this expression will be evaluated at the current simulation time. Note the signal attribute and array constant specified in the expression.

  examine -expr {clk’event && (/top/xyz == 16’hffae)}

  Commands like find and **examine** return their results as a Tcl list (just a blank-separated list of strings). You can do things like:

  foreach sig [find sig ABC*] {echo "Signal $sig is [exa $sig]" …}

  if {[examine -bin signal_12] == “11101111XXXZ”} {…}

  examine -hex [find *]

  The Tcl variable array, **$examine ()**, can also be used to return values. For example, $examine (/clk). You can also examine an object in the Source Window by selecting it with the right mouse button.

- Print the value of C variable *x*.

  examine x

- Print the value *p (de-references *p*).

  examine *p

- Print the structure member *in1* pointed to by *ip*.

examine ip->in1

Related Topics

- Design Object Names
- Wildcard Characters
- DefaultRadix *modelsim.ini* variable
- GUI_expression_format
- Preserving Object Visibility for Debugging Purposes
- C Debug

exit

This command exits the simulator and the Questa SIM application.

If you want to stop the simulation using a when command, use a stop command within your when statement, do not use an exit command or a quit command. The stop command acts like a breakpoint at the time it is evaluated.

Syntax

exit [-force] [-code <integer>]

Arguments

- -force

  (optional) Quits without asking for confirmation. If this argument is omitted, Questa SIM asks you for confirmation before exiting. You can also use -f as an alias for this switch.

- -code <integer>

  (optional) Quits the simulation and issues an exit code.

  <integer> — This is the value of the exit code. You should not specify an exit code that already exists in the tool. Refer to the section "Exit Codes" in the User's Manual for a list of existing exit codes. You can also specify a variable in place of <integer>.

  You should always print a message before executing the exit -code command to explicitly state the reason for exiting.

Examples

You can use exit -code to instruct a vmake command to exit when it encounters an assertion error. The onbreak command can specify commands to be executed upon an assert failure of sufficient severity, after which the simulator can be made to return an exit status. This is shown in the following example:

```
set broken 0
onbreak {
  set broken 88
  resume
}
run -all
if { $broken } {
  puts "failure -- exit status $broken"
  exit -code $broken
} else {
  puts "success"
}
quit -f
```

The resume command gives control back to the commands following the run -all to handle the condition appropriately.

fcover configure

This command enables, disables, and sets coverage targets for SystemVerilog and PSL cover directives.

If you invoke it without any optional arguments, the command reports the current settings for the specified directive(s).

Syntax

fcover configure **<coverage_directive>...** [-at_least <count>] [-disable] [-enable] [-exclude] [-include] [-limit <count>] [-log on | off] [-recursive] [-weight <integer>]

Arguments

- <coverage_directive>...

  (required) Identifies the functional coverage directive(s) to which the configuration parameters should be applied. Multiple directives may be specified. Wildcards are allowed.

- -at_least <count>

  (optional) Specifies a target count for the selected coverage point(s).

  <count> — Any positive integer where the default value is 1.

  A directive is considered 100% covered when the specified count is reached. You can change the permanent default for this argument by editing the CoverAtLeast variable in the *modelsim.ini* file.

- -disable

  (optional) Disables incrementing on the specified directive(s). Disabled directives still count toward overall coverage if they had coverage events prior to being disabled. You can change the permanent default for this argument by editing the CoverEnable variable in the *modelsim.ini* file.

- -enable

  (optional) Enables incrementing on the specified directive(s). You can change the permanent default for this argument by editing the CoverEnable variable in the *modelsim.ini* file. (default)

- -exclude

  (optional) Removes the specified directive(s) from the current functional coverage database. Excluded directives will still show up in the Cover Directives window but they do not count toward coverage totals nor do they show up in reports.

- -include

  (optional) Adds the specified directive(s) to the current functional coverage database. (default)

- -limit <count>

  (optional) Specifies the number of SystemVerilog or PSL cover directive hits before the directive is auto disabled.

  <count> — Any positive integer greater than or equal to the value of -at_least, where the default is -1 (unlimited).

  The value for -limit is automatically adjusted to equal the value for -at_least if the value for -at_least is set to a greater value than the value for -limit.You can change the permanent default for this argument by editing the CoverLimit variable in the *modelsim.ini* file.

  This argument is useful if you have a directive that is already covered but is evaluated frequently. Disabling such directives improves simulation performance.

- -log on | off

  (optional) Specifies whether to log cover directive messages to the transcript. Even with logging off, coverage messages are incremented in the underlying database.This option applies to all cover directives that follow it in the command line.

  One of the following values is required:

  on — Enable logging. (default)

  off — Disable logging.

  You can change the permanent default for this argument by setting the CoverLog variable in the *modelsim.ini* file.

- -recursive

  (optional) For use with wildcard matching. Specifies that the path of the matching search is to descend recursively into subregions. If omitted, the search is limited to the selected region.

- -weight <integer>

  (optional) Specifies a relative weighting for the specified coverage directive.

  <integer> — Any natural integer (>= 0) where the default is 1.

  A directive with a weight of 2 will have twice the impact on the aggregated coverage numbers than a directive with a weight of 1. If its weight is set to 0 but a directive is otherwise enabled, its coverage will not count toward the aggregated coverage statistics.

Examples

- Report the current configuration settings for the functional coverage directives in the current scope.

  fcover configure *

- Enable coverage counting for */top/reset_trigger* with an -at_least count of 20.

  fcover configure -enable -at_least 20 /top/reset_trigger

- Disable all functional coverage directives in the region */top* and all regions below */top*.

```
fcover configure -r -disable /top/*
```

Related Topics

- Changing the Default Configuration of Cover Directives
- CoverAtLeast
- CoverEnable
- CoverLimit
- CoverLog
- Verification with Functional Coverage
- Verification with Assertions and Cover Directives

find

This command locates objects by type and name. Arguments to the command are grouped by object type.

Syntax

find nets | signals **<object_name> …** [-internal] [-nofilter] {[-in] [-inout] [-out] | [-ports]} [-recursive]

find instances | blocks **{<object_name> …** | -bydu <design_unit> | -file <file_name>**}** [-arch] [-recursive] [-nodu]

find virtuals **<object_name> …** [-kind <kind>] [-unsaved] [-recursive]

find classes [<class_name>]

find objects [-class <class_name>] [-isa <class_name>] [<object_name>]

Arguments

Arguments to the command are grouped by object type.

- Arguments for nets and signals
- Arguments for instances and blocks
- Arguments for virtuals
- Arguments for classes
- Arguments for objects

Arguments for nets and signals

When searching for nets and signals, the find command returns the full pathname of all nets, signals, registers, variables, and named events that match the name specification.

- <object_name> …

  (required) Specifies the net or signal for which you want to search. Multiple nets and signals and wildcard characters are allowed. Wildcards cannot be used inside of a slice specification. Spaces, square brackets, and extended identifiers require special syntax; see the examples below for more details.

- -in

  (optional) Specifies that the scope of the search is to include ports of mode IN.

- -inout

  (optional) Specifies that the scope of the search is to include ports of mode INOUT.

- -internal

  (optional) Specifies that the scope of the search is to include internal (non-port) objects.

- -nofilter

  (optional) Specifies that the *WildcardFilter* Tcl preference variable be ignored when finding signals or nets.

- -out

  (optional) Specifies that the scope of the search is to include ports of mode OUT.

- -ports

  (optional) Specifies that the scope of the search is to include all ports. Has the same effect as specifying -in, -out, and -inout together.

- -recursive

  (optional) Specifies that the scope of the search is to descend recursively into subregions. If omitted, the search is limited to the selected region.

Arguments for instances and blocks

When searching for instances, the find command returns the primary design unit name.

- -arch

  Used with “instances” only: Lists the corresponding architecture name along with the entity name for any VHDL design unit names returned by the find command.

- -bydu <design_unit>

  Searches for a design unit. Mutually exclusive with -file and <object_name>.

  <design_unit> — Name of a single design unit to search for. This argument matches the pattern specified by primary <design_unit> of the instance only. Library and Secondary names are not supported.

- -file <file_name>

  Writes a complete list of the instances in a design to a file. Mutually exclusive with -bydu and <object_name>.

  <file_name> — A string specifying the name for a file.

- <object_name> …

  Specifies the name of an instance or block for which you want to search. Multiple instances and wildcard characters are allowed. Mutually exclusive with -file and -bydu.

- -recursive

  (optional) Specifies that the scope of the search is to descend recursively into subregions. If omitted, the search is limited to the selected region.

- -nodu

  (optional) Removes the "du" string from the names of design units found with -bydu argument.

Arguments for virtuals

When searching for virtuals, all optional arguments must be specified before any object names.

- <object_name> …

  (required) Specifies the virtual object for which you want to search. Multiple virtuals and wildcard characters are allowed.

- -kind <kind>

  (optional) Specifies the kind of virtual object for which you want to search.

  <kind> — A virtual object of one of the following kinds:

  - designs
  - explicits
  - functions
  - implicits
  - signals.

- -unsaved

  Specifies that Questa SIM find only virtuals that have not been saved to a format file.

Arguments for classes

- <class_name>

  (optional) Specifies the incrTcl class for which you want to search. Wildcard characters are allowed. The options for class_name include nets, objects, signals, and virtuals. If you do not specify a class name, the command returns all classes in the current namespace context. See incrTcl commands in the Tcl Man Pages (Help > Tcl Man Pages) for more information.

Arguments for objects

- -class <class_name>

  (optional) Restricts the search to objects whose most-specific class is class_name.

- -isa <class_name>

  (optional) Restricts the search to those objects that have class_name anywhere in their heritage.

- <object_name>

  (optional) Specifies the incrTcl object for which you want to search. Wildcard characters are allowed. If you do not specify an object name, the command returns all objects in the current namespace context. See incrTcl commands in the Tcl Man Pages (Help > Tcl Man Pages) for more information.

Description

The following rules are used by the find command to locate an object:

- If the name does not include a dataset name, then the current dataset is used.
- If the name does not start with a path separator, then the current context is used.

- If the name is a path separator followed by a name that is not the name of a top-level design unit, then the first top-level design unit in the design is used.
- For a relative name containing a hierarchical path, if the first object name cannot be found in the current context, then an upward search is done up to the top of the design hierarchy to look for a matching object name.
- If no objects of the specified name can be found in the specified context, then an upward search is done to look for a matching object in any visible enclosing scope up to an instance boundary. If at least one match is found within a given context, no (more) upward searching is done; therefore, some objects that may be visible from a given context will not be found when wildcards are used if they are within a higher enclosing scope.
- The wildcards '*' and '?' can be used at any level of a name except in the dataset name and inside of a slice specification. Square bracket ([]) wildcards can also be used.
- A wildcard character will never match a path separator. For example, */dut/** will match */dut/siga* and */dut/clk*. However, */dut** won't match either of those.
- Because square brackets are wildcards in the find command, only parentheses (()) can be used to index or slice arrays.
- The *WildcardFilter* Tcl preference variable is used by the find command to exclude the specified types of objects when performing the search.

See Design Object Names for more information on specifying names.

Examples

- Find all signals in the entire design.

  find signals -r /*

- Find all instances in the entire design and save the list in the file *instancelist.txt*.

  find instances -file instancelist.txt -r /*

- Find all input signals in region /top that begin with the letters "xy".

  find nets -in /top/xy*

- Find all signals in the design hierarchy at or below the region *<current_context>/u1/u2* whose names begin with "cl".

  find signals -r u1/u2/cl*

- Find a signal named *s1*. Note that you must enclose the object in braces because of the square bracket wildcard characters.

  find signals {s[1]}

- Find signals *s1*, *s2*, or *s3*.

  find signals {s[123]}

- Find the element of signal *s* that is indexed by the value 1. Note that the find command uses parentheses (()), not square brackets ([]), to specify a subelement index.

  find signals s(1)

- Find a 4-bit array named *data*. Note that you must use braces ({}) due to the spaces in the array slice specification.

  find signals {/top/data(3 downto 0)}

- Note that when specifying an object that contains an extended identifier as the last part of the name, there must be a space after the closing '\' and before the closing '}'.

  find signals {/top/\My extended id\ }

- If */dut/core/pclk* exists, prints the message "pclk does exist" in the transcript. This would typically be run in a Tcl script.

  if {[find signals /dut/core/pclk] != ""} {
  echo "pclk does exist"

- Find instances based on their names using wildcards. Send search results to a text file that lists instance names, including the hierarchy path, on separate lines.

```
# Search for all instances with u1 in path
set pattern_match "*u1*" ;

# Get the list of instance paths
set inst_list [find instances -r *]  ;

# Initialize an empty list to strip off the architecture names
set ilist [list]  ;

foreach inst $inst_list {
set ipath [lindex $inst 0]
 if {[string match $pattern_match $ipath]} {
 lappend ilist $ipath
 }
}
# At this point, ilist contains the list of instances only--
# no architecture names
#
# Begin sorting list
set ilist [lsort -dictionary $ilist]

# Open a file to write out the list
set fhandle [open "instancelist.txt" w]
foreach inst $ilist {
 # Print instance path, one per line
 puts $fhandle $inst
}

# Close the file, done.
close $fhandle ;
```

Additional search options

To search for HDL objects within a specific display window, use the search command or select **Edit** > **Find**.

Related Topics

- Design Object Names
- Wildcard Characters

find connections

This command returns the set of nets that are electrically equivalent to a specified net. It is only available during a live simulation.

Syntax

find connections <net>

Arguments

- <net>

  (required) A net in the design. Returns a list of nets connected to the specified net. For example:

  find connections /top/p/strb

  returns:

```
# Connected nets for strb
#    output : /top/p/strb
#  internal : /top/pstrb
#     input : /top/c/pstrb
```

find drivers

Prerequisite:

- Use the -debugdb switch with the vopt command.

Or,

- If vopt is not used, simulate the design with the -debugdb and -voptargs=”+acc” arguments to the vsim command.

This command traces backward in time to find the active driver(s), processes, or first elements of the specified signal or signal event. Processes may be combinatorial or sequential assignments. The command can trace through multiple clock cycles and multiple clock domains. Traces are executed either during simulation or post-simulation.

You can do the following types of causality traces:

- Show Cause — Trace back to the first sequential element causing the event. Duplicates the GUI menu paths: Show Cause > Show Cause or Show Cause > Show Cause from Time. (See Initiating Causality Traceback from the GUI.)
- Show Driver — Trace the specified signal back to the immediate driving process(es). The process(es) may be combinatorial or sequential assignments. Duplicates the GUI menu paths: Show Cause > Show Driver or Show Cause > Show Driver from Time.
- Show Root Cause — Trace the event back as far as possible. This may cross multiple clock cycles and multiple clock domains. Duplicates the GUI menu paths: Show Cause > Show Root Cause or Show Cause > Show Root Cause from Time.
- Show All Possible Drivers — Display all possible driving assignments for a signal (does not use the simulation results to determine.) Duplicates the GUI menu path: Show Cause > Show All Possible Drivers.

Note

Traces may find more than one active driver for the specified signal and time. To continue backtracing, you must start a new search on one of the drivers identified in the Multiple Drivers dialog or the transcript. You must also specify the time that the signal was found. Refer to the Example section for more information.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

You can interrupt the various "find drivers" operations by using the Escape key.

A help menu is also available by entering find drivers -help or find drivers -?

Note

The Show Cause and Show Root Cause options will display a warning if you simulate with the -novopt switch. You will only have access to the Show Driver option in the Source Window. Full causality traceback functionality requires optimization of your design with vopt or vsim -voptargs. Refer to Using Causality Traceback for more information.

Syntax

Show driver

find drivers [-schematic] [-last] [-source] [-transcript [-compact <string> | -tcl] [-noclip | -width <n>]] [**-time <time> [unit]**] [-wave] **<signal>**

Show cause

find drivers **-cause** [-schematic] [-last] [-source] [-transcript [-compact <string> | -tcl] [-noclip | -width <n>] [**-time <time> [unit]**] [-wave] **<signal>**

Show root cause

find drivers **-rootcause** [-schematic] [-last] [-source] [-transcript [-compact <string> | -tcl] [-noclip | -width <n>] [**-time <time> [unit]**] [-wave] **<signal>**

Show root cause of 'X'

find drivers **-chasex** [-schematic] [-last] [-source] [-transcript [-compact <string> | -tcl] [-noclip | -width <n>] [**-time <time> [unit]**] [-wave] **<signal>**

Show all possible drivers

find drivers **-possible** [-schematic] [-last] [-transcript [-compact <string> | -tcl] [-noclip | -width <n>] [-wave] **<signal>**

Arguments

- -cause

  (optional) Traces the specified <signal> back to the first sequential element causing the event.

- -chasex

  (optional) Traces to the root cause of an 'X' on the specified signal over multiple cycles and clock domains.

- -compact <string>

  (optional) Specifies a compact format for data output, using a specified text string to separate the fields.

  <string> — Any combination of characters. Special characters are allowed except a backslash (\ ).

- -schematic

  (optional) All signals contained in the path found by the trace are placed in a dedicated Schematic window.

- -last

  (optional) Returns the results from the last completed trace to the transcript. Useful for trying the various format options. Allows you to quickly see how each format option (-compact, -tcl, -width, and -noclip) affects the output.

- -noclip

  (optional) Allows columns to be arbitrarily long when returned to the transcript.

- -possible

  (optional) Display all of the possible driving assignments for a signal (simulation results are not used). The data returned is hyperlinked to the driving signal in the source file.

- -rootcause

  (optional) Trace the event back as far as possible. This may cross multiple clock cycles and multiple clock domains.

- -source

  (optional) Opens the source file containing the line of code found by the trace and scrolls to show that line as well as highlight the driving signal. If the trace type is -possible then this option is not allowed.

- -tcl

  (optional) Displays trace results in a TCL list.

- -time <time> [unit]

  (optional) Traces <signal> from a specified time. If not specified, the trace begins from the current active time indicated by either the end of the simulation, or the Wave window's active cursor. The -time option is not allowed when using the -possible argument.

  <time> — Specified as an integer or decimal number. Current simulation units are the default unless specifying <unit>.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid VHDL time units are: fs, ps, ns, us, ms, sec, min, and hr. You must enclose <time> and <unit> within braces ({}).

- -transcript

  (optional) Specifies that trace results are reported to the Transcript window in tabular format (unless the -compact argument is used). Default behavior if not using -schematic, -source, or -wave. Returns the following information:

  "Causality trace from the time <time> for the signal <signal>"

  - Time — Simulation time when the driving signal caused a change in the traced signal.

- Type — Register containing the traced signal.
- Scope — Name of the signal that caused the change in the signal being traced.
- Source File — Location of the signal in the source file. Includes the full path, name of the source file, and line number. The text is hyperlinked to the location of the driving signal. Double-clicking the entry opens the source and highlights the source code.

- -wave

  (optional) Specifies that all signals in the path found by the trace are added to a dedicated Wave window, and cursors are added that show the beginning and ending times of the trace. The dedicated Wave window is cleared of signals before displaying the results of a new trace.

- -width <n>

  (optional) Specifies the maximum size of each column when data is returned to the transcript in tabular form.

  <n> — Any positive integer.

- <signal>

  (required) The path name of a signal in the Wave window. Must be specified as the final argument to the find drivers command.

Examples

- Show the first sequential process for the signal test1/jr

  find drivers -source -time {698 ns} -cause -transcript sim:/test1/jr

  Locate the driving signal for the Flip Flop /test1/ff1 in the source file by double clicking on the linked text:

  `C:\trace/test1.sv:106.`

  The driving signal is highlighted in the source file.

- Trace a signal that finds Multiple Active Drivers.

  find drivers -source -time {867 ns} -cause -transcript sim:/test1/mnf

  Returns:

  ```
  # Multiple active drivers exist at time 822 ns for signal
  sim:/test1/mnf:
  #    PROCESS: #ASSIGN#33  SIGNAL: /test1/ar  FILE:
  C:\ctraceback/test1.sv:33
  #    PROCESS: #ASSIGN#33  SIGNAL: /test1/br  FILE:
  C:\ctraceback/test1.sv:33
  ```

  To continue the trace you must select one of the multiple drivers and execute the find drivers command again on that signal, for example:

  find drivers -source -time {822 ns} -cause -transcript /test1/br

Returns:

```
# Causality trace from time 822 ns for signal "sim:/test1/br":
#
#   Time | Type | Scope   | Signal | Source File
# ------ | ---- | ------  | ------ | ----------------------------
# 822 ns | Gate | /test1  |        | C:\ctraceback/test1.sv:66
```

Related Topics

- Using Causality Traceback

find infiles

This command searches for a string in the specified file(s) and prints the results to the Transcript window. The results are individually hotlinked and will open the file and display the location of the string.

When you execute this command in command-line mode from outside of the GUI, the results are sent to stdout with no hotlinks.

Arguments to this command are order dependent. Follow the order specified in the Syntax section.

Syntax

find infiles **<string_pattern> <file>...**

Arguments

- <string_pattern>

  (required) The string you are searching for. You can use Tcl regular expression wildcards to further restrict the search capability.

- <file>...

  (required) The file(s) to search. You can use Tcl regular expression wildcards to further restrict the search capability.

Examples

Figure 2-2 shows a screen capture containing a few examples of the find infiles command and the results.

Figure 2-2. find infiles Example

find insource

This command searches for a string in the source files for the current design and prints the results to the Transcript window. The results are hotlinked individually and will open the file and display the location of the string.

When you execute this command in command-line mode from outside of the GUI, the results are sent to stdout with no hotlinks.

Syntax

find insource <pattern> [-exact | -glob | -regexp] [-inline] [-nocase]

Arguments

- <pattern>

  (required) The string you are searching for. You can use regular expression wildcards to further restrict the search. You must enclose <pattern> in quotes (“ “) if it includes spaces. You must specify the <pattern> at the end of the command line; any switches specified after <pattern> will not be registered.

- -exact | -glob | -regexp

  (optional) Defines the style of regular expression used in the <pattern>

  -exact — Indicates that no characters have special meaning, thus disabling wildcard features.

  -glob — (default) Allows glob-style wildcard characters. For more information refer to the Tcl documentation:

  Help > Tcl Man Pages

  Select “Tcl Commands”, then “string”, then “string match”

  -regexp — Allows Tcl regular expressions. For more information refer to the Tcl documentation:

  Help > Tcl Man Pages

  Select “Tcl Commands”, then “re_syntax”.

- -inline

  (optional) Returns the matches in the form of a Tcl list, which disables the hotlink feature but allows for easier post-processing.

- -nocase

  (optional) Treats <pattern> as case-insensitive.

Examples

- Figure 2-3 shows a couple of examples of the find insource command and the results.

Figure 2-3. find insource Example

- Searching for two keywords with whitespace between them:

  ```
  find insource -regexp {top_dut\s+dut}
  ```

  returns:

  ```
  # top.sv:20:        top_dut                 dut (
  ```

- Searching for string starting with 'dut' and ending with 'o':

  ```
  find insource -regexp {dut.*o}
  ```

  returns:

  ```
  # top.sv:17:        DUT_io dut_io(.clock(tb_clk), .reset(tb_reset));
  ```

- Searching for string irrespective of case:

  ```
  find insource -regexp -nocase {DUT}
  ```

  returns:

  ```
  # test.sv:10:       virtual DUT_io dut_io;
  # test.sv:27:       this.dut_io = dut_io;
  ```

Related Topics

typespec

DISABLE_ELAB_DEBUG environment variable

force

This command allows you to apply stimulus interactively to VHDL signals, Verilog nets and registers, and SystemC boundary types.

It is possible to create a complex sequence of stimuli when the force command is included in a DO file.

There are a number of constraints on what you can and cannot force.

You can force:

- VHDL signals or parts of signals.
- Verilog nets and registers, bit-selects, part-selects, and field-selects. Refer to "Force and Release Statements in Verilog" for more information.
- "Virtual Signals" if the number of bits corresponds to the signal value.
- An alias of a VHDL signal.
- Signals within SystemC modules, with the following limitations:

  Only mixed language boundary types are supported in mixed language simulations.

  The -drive option to force is not supported.

  Individual bits and slices may not be forced or unforced.
- An input port that is mapped at a higher level in VHDL and mixed models.

You cannot force:

- Virtual functions.
- VHDL variables. Refer to the change command for information on working with VHDL variables.
- An input port that has a conversion function on the input or on the path up the network mapped from the input.

This command provides additional information with the -help switch.

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

Forcing values, driver type, repetition time or stop time on an object

force **{<object_name> <value>** [[@]<time_info>][, <value> [@]<time_info>]...
[-deposit | -drive | -freeze] [-cancel [@]<time_info>] [-repeat [@]<time_info>]

Reporting all force commands

If you specify this command without arguments, it returns a list of the most recently applied force commands and a list of forces coming from the Signal Spy signal_force() and $signal_force() calls from within VHDL, Verilog, and SystemC source code.

For example, after executing:

force -freeze /top/p/addr 0 100, 1 150 -r 200 -cancel 2000

the times specified are relative to the current simulation time, in this case 2820 ns

entering:

force

returns:

```
# force -freeze /top/p/addr 0 {@2920 ns} , 1 {@2970 ns}
   -repeat {@3020 ns} -cancel {@4820 ns}
```

Note

The simulator translates the relative time you specify into absolute time when the force command is executed.

Arguments

- <object_name>

  (required when forcing a value change) Specifies the name of the HDL object to be forced. A wildcard is permitted only if it matches one object. Refer to Design Object Names and Tcl Syntax and Specification of Array Bits and Slices for the full syntax of an object name. The object name must specify a scalar type or a one-dimensional array of character enumeration. You may also specify a record sub-element, an indexed array, or a sliced array, as long as the type is one of the above. Must be specified as the first argument to the **force** command.

- <value>

  (required when forcing a value change) Specifies the value to which the object is to be forced. The specified value must be appropriate for the type. Must be specified as the second argument to the force command.

  A one-dimensional array of character enumeration can be forced as a sequence of character literals or as a based number with a radix of 2, 8, 10 or 16. For example, the following values are equivalent for a signal of type bit_vector (0 to 3):

Description	VHDL Value	Verilog Value
character literal sequence	F	F
binary radix	2#1111	'b1111
octal radix	8#17	'o17
decimal radix	10#15	'd15

hexadecimal radix	16#F	'hF

Note

For based numbers in VHDL, Questa SIM translates each 1 or 0 to the appropriate value for the number's enumerated type. The translation is controlled by the translation table in the *pref.tcl* file. If Questa SIM cannot find a translation for 0 or 1, it uses the left bound of the signal type (type'left) for that value.

You can create a sequence of forced values on an object by specifying <value> [@]<time_info> in a comma/space separated list.

For example:

```
force /top/p/addr 1 100ns, 0 200ns, 1 250ns
```

- -cancel [@]<time_info>

  (optional) Cancels the force command at the time specified by <time_info>.

  where:

  <time_info> is [@]<time_value>[<time_unit>]

  Refer to [@]<time_info> for more information about specifying time values.

- -drive

  (optional) Attaches a driver to the object and drives the specified <value> until the object is forced again or until it is unforced with the noforce command.

  This option is illegal for unresolved signals or SystemC signals.

- -deposit

  (optional) Sets the object to the specified <value>. The <value> remains until the object is forced again, there is a subsequent driver transaction, or it is unforced with a noforce command.

Note

If the -freeze, -drive, or -deposit options are not used, then -freeze is the default for unresolved objects, and -drive is the default for resolved objects. If you prefer -freeze as the default for resolved and unresolved VHDL signals, change the DefaultForceKind variable in the *modelsim.ini* file.

- -freeze

  (optional) Freezes the object at the specified <value> until it is forced again or until it is unforced with the noforce command.

Note

If you prefer -freeze as the default for resolved and unresolved VHDL signals, change the DefaultForceKind variable in the *modelsim.ini* file.

- -repeat [@]<time_info>

  (optional) Repeats a series of forced values and times at the time specified.

  where:

  <time_info> is [@]<time_value>[<time_unit>]

  Refer to [@]<time_info> for more information about specifying time values.

  You must specify at least two <value> <time_info> pairs on the forced object before specifying -repeat, for example:

  force top/dut/p 1 0, 0 100 -repeat 200 -cancel 1000

  A repeating force command will force a value before other non-repeating force commands that occur in the same time step.

- [@]<time_info>

  (optional) Specifies the relative or absolute simulation time at which the <value> is to be applied.

  where:

  <time_info> is [@]<time_value>[<time_unit>]

  @ — A prefix applied to <time_value> to specify an absolute time. By default, the specified time units are assumed to be relative to the current time unless the value is preceded by the character "at" (@). Omit the "at" (@) character to specify relative time. For example:

  ```
  -cancel {520 ns}      \\ Relative Time
  -cancel {@ 520 ns}    \\ Absolute Time
  ```

  <time_value> — The time (either relative or absolute) to apply to <value>. Any non-negative integer. A value of zero cancels the force at the end of the current time period.

  <time_unit> — An optional suffix specifying a time unit where the default is to use the current simulator time by omitting <time_unit>. Valid time units are: fs. ps, ns, us, ms, sec, min, and hr.

  <time_value> and <time_unit> can be formatted in any of the following ways:

  ```
  10ns
  10 ns
  {10 ns}
  “10 ns”
  ```

Note

If you specify a sequence of forces and use braces ({}) surrounding a <time_value> and <time_unit> pair, you must place a space in front of the comma (,) separating the two value/time pairs. For example:

```
force foo 1 {10 ns} , 0 {20 ns}
```

Examples

- Reporting all recently applied force commands

If you specify this command with no arguments, it returns a list of all forced objects and the changes applied. For example, after executing:

force -freeze /top/p/addr 0 100, 1 150 -r 200 -cancel 2000

where the times specified are relative to the current simulation time, in this case 2820 ns.

Entering:

force

returns:

```
# force -freeze /top/p/addr 0 {@2920 ns} , 1 {@2970 ns}
   -repeat {@3020 ns} -cancel {@4820 ns}
```

Note

Executing the force command translates the relative time you specified into absolute time.

- Force *input1* to 0 at the current simulator time.

  force input1 0

- Force the fourth element of the array *bus1* to 1 at the current simulator time.

  force bus1(4) 1

- Force *bus1* to 01XZ at 100 nanoseconds after the current simulator time.

  force bus1 2#01XZ 100 ns

- Force *bus1* to 16#F at the absolute time 200 measured in the resolution units selected at simulation start-up.

  force bus1 16#f @200

- Force *input1* to 1 at 10 time units after the current simulation time and to 0 at 20 time units after the current simulation time. Repeat this cycle every 100 time units after the current simulation time, If the current simulation time is 100 ns, the next transition is to 1 at 110 ns and 0 at 120 ns, this pattern to start repeating at 200 ns.

  force input1 1 10, 0 20 -r 100

- Similar to the previous example, but also specifies the time units.

  force input1 1 10 ns, 0 20 ns -r 100 ns

- Force signal *s* to alternate between values 1 and 0 every 100 time units until 1000 time units have occurred, starting from time Now. Cancellation occurs at the last simulation delta cycle of a time unit.

  force s 1 0, 0 100 -repeat 200 -cancel 1000

So,

```
force s 1 0 -cancel 0
```

will force signal *s* to 1 for the duration of the current time period.

- Force *siga* to decimal value 85 whenever the value on the signal is 1.

```
when {/mydut/siga = 10#1} {
  force -deposit /mydut/siga 10#85
}
```

- Force one bit of a record containing an array.

```
force struct1.bus1(4) 1
```

- Force a slice of an array.

```
force {bus1[2:5]} 'hF
```

Related Topics

- change
- DefaultForceKind
- Design Object Names
- Force and Release Statements in Verilog
- Force Command Defaults
- noforce
- Virtual Signals

formatTime

This command provides global format control for all time values displayed in the GUI. When specified without arguments, this command returns the current state of the three arguments.

Syntax

formatTime [[+|-]commas] [[+|-]nodefunits] [[+|-]bestunits]

Arguments

- [+|-]commas

  (optional) Insert commas into the time value.

  + prefix — On

  - prefix — Off. (default)

- [+|-]nodefunits

  (optional) Do not include default unit in the time.

  + prefix — On

  - prefix — Off. (default)

- [+|-]bestunits

  (optional) Use the largest unit value possible.

  + prefix — On

  - prefix — Off. (default)

Examples

- Display commas in time values.

  formatTime +commas

  Instead of displaying 6458131 ps, the GUI will display 6,458,131 ps.

- Use largest unit value possible.

  formatTime +bestunits

  Displays 8 us instead of 8,000 ns.

fsm list

This command returns information about recognized finite state machines, including the number of states and transitions.

The output matches that of the FSM List window.

Syntax

fsm list

Arguments

None

Examples

fsm list

Returns:

```
{sim:/top/A/I0/present_state 10 165} {sim:/top/A/I1/present_state 7 98}
{sim:/top/B/present_state 3 18} {sim:/top/B/I0/present_state 10 165}
{sim:/top/B/I1/present_state 14 302}
```

which follows the form:

```
{<dataset>:<instance> <number_of_states> <number_of_transitions>}
```

Related Topics

- fsm view
- fsm properties
- Finite State Machines
- FSM List Window
- FSM Viewer Window

fsm properties

This command returns information about the specified finite state machine.

This information matches that found in the FSM Properties dialog box.

Syntax

fsm properties <fsm_instance>

Arguments

- <fsm_instance>

  (required) You can produce a list of available fsm instances with the fsm list command.

Examples

fsm properties /top/A/I0/present_state

Returns:

```
top/A/I0/clk {RESET 0 S0 1 S1 2 S2 3 S3 4 S4 5 S5 6 S6 7 S7 8 S8 9} RESET
{top/A/I0/#ALWAYS#53 top/A/I0/#ALWAYS#60} {top/A/I0/clk top/A/I0/rst
top/A/I0/next_state top/A/I0/enable} {top/A/I0/present_state top/A/I0/en2
top/A/I0/en1 top/A/I0/en0 top/A/I0/dval top/A/I0/fifo_rd}
```

which follows the form:

```
<clk> {<state_encoding>} <reset_state> {process_list} {<clk> <inputs>
<outputs>}
```

Related Topics

- fsm view
- fsm list
- FSM List Window
- FSM Viewer Window
- Finite State Machines

fsm view

This command displays the specified finite state machine in the FSM List window.

Syntax

fsm view <fsm_instance>

Arguments

- <fsm_instance>

  (required) You can produce a list of available fsm instances with the fsm list command.

Examples

fsm view /top/A/I0/present_state

Returns:

```
{sim:/top/A/I0/present_state 10 165} {sim:/top/A/I1/present_state 7 98}
{sim:/top/B/present_state 3 18} {sim:/top/B/I0/present_state 10 165}
{sim:/top/B/I1/present_state 14 302}
```

which follows the form:

```
{<dataset>:<instance> <number_of_states> <number_of_transitions>}
```

Related Topics

- fsm list
- fsm properties
- FSM List Window
- FSM Viewer Window
- Finite State Machines

gc configure

Prerequisite:

- Regular simulation — Simulate without specifying -classdebug with vsim.
- Interactive class debugging — Simulate with vsim -classdebug.

This command specifies when the System Verilog garbage collector will run. The garbage collector may be configured to run after a memory threshold has been reached, after each simulation run command completes, and/or after each simulation step command. The default settings are optimized to balance performance and memory usage for either regular simulation or class debugging (vsim -classdebug). Returns the current settings when specified without arguments.

Syntax

gc config [-onrun 0 | 1] [-onstep 0 | 1] [-threshold <n>]

Arguments

- -onrun 0 | 1

  (optional) Enables/disables garbage collector execution after each simulation run command completes.

  0 — Off, default for regular simulation

  1 — On, default for interactive class debugging

- -onstep 0 | 1

  (optional) Enables or disables garbage collector execution after each step when stepping through your simulation.

  0 — Off, default for both regular simulation and interactive class debugging.

  1 — On

- -threshold <n>

  (optional) Sets the maximum amount of memory in megabytes allocated for storage of class objects before the garbage collector runs.

  <n> — Any positive integer where <n> is the number of megabytes.

  Regular simulation default =100 megabytes

  Interactive class debugging default = 5 megabytes

Related Topics

- gc configure
- vsim -classdebug
- vsim -gconrun
- vsim -gconstep
- vsim -threshold
- SystemVerilog Class Debugging
- Class Instance Garbage Collection
- GCThreshold
- GCThresholdClassDebug

gc run

This command runs the SystemVerilog garbage collector.

Syntax

gc run

Arguments

None

Related Topics

- gc configure
- vsim -gconrun
- vsim -gconstep
- vsim -threshold
- SystemVerilog Class Debugging
- Class Instance Garbage Collection
- GCThreshold
- GCThresholdClassDebug

gdb dir

This command sets the source directory search path for the C debugger and starts the C debugger if it is not already running.

Syntax

gdb dir [<src_directory_path_1>...]

Arguments

- <src_directory_path_1>...

  (optional) Specifies one or more directories for C source code. If no directory is specified, the source directory search path is set to the gdb default–$cdir:$cwd. Either absolute or relative paths may be used. Specify multiple paths as a space separated list. Wildcards and relative paths are allowed.

Examples

Set the source directory search paths to ../dut/ and ../foo/.

```
gdb dir ../dut/ ../foo/
```

Related Topics

- C Debug
- Setting Up C Debug

getactivecursortime

This command gets the time of the active cursor in the Wave window and returns the time value.

Syntax

getactivecursortime [-window <wname>]

Arguments

- -window <wname>

  (optional) Specifies an instance of the Wave window that is not the default.

  <wname> — The name of the window that is not the default.

  Use the view command to change the default window.

Examples

getactivecursortime

Returns:

```
980 ns
```

Related Topics

- left
- right

getactivemarkertime

This command gets the time of the active marker in the List window.

Returns the time value. If -delta is specified, returns time and delta.

Syntax

getactivemarkertime [-window <wname>] [-delta]

Arguments

- -window <wname>

  (optional) Specifies an instance of the List window that is not the default. Otherwise, the default List window is used.

  <wname> — The name of the window that is not the default.

  Use the view command to change the default window.

- -delta

  (optional) Returns the delta value where the default is to return only the time.

Examples

getactivemarkertime -delta

Returns:

```
980 ns, delta 0
```

Related Topics

- down
- up

help

This command displays in the Transcript window a brief description and syntax for the specified command.

Syntax

help [<command> | <topic>]

Arguments

- <command>

  (optional) Specifies the command for which you want help. The entry is case and space sensitive.

- <topic>

  (optional) Specifies a topic for which you want help. The entry is case and space sensitive. Specify one of the following six topics:

Topic	Description
commands	Lists all available commands and topics
debugging	Lists debugging commands
execution	Lists commands that control execution of your simulation.
Tcl	Lists all available Tcl commands.
Tk	Lists all available Tk commands
incrTCL	Lists all available incrTCL commands

history

This command lists the commands you have executed during the current session. History is a Tcl command. For more information, consult the Tcl Man Pages (Help > Tcl Man Pages).

Syntax

history [clear] [keep <value>]

Arguments

- clear

  (optional) Clears the history buffer.

- keep <value>

  (optional) Specifies the number of executed commands to keep in the history buffer.

  <value> — Any positive integer where the default is 50.

jobspy

This command controls JobSpy, a tool for monitoring and controlling batch simulations and simulation farms.

This command provides additional information with the -help switch.

Syntax

jobspy [-gui | -startd | -killd | jobs | status | <jobid> <command>]

Arguments

- -gui

  (optional) Launches the JobSpy Job Manager GUI.

- -startd

  (optional) Starts the jobspy daemon which enables job tracking. You must first set the JOBSPY_DAEMON environment variable before starting the daemon.

  You must specify this switch if you set the JOBSPY_DAEMON to a port@host.

  You do not need to specify this switch if you set the JOBSPY_DAEMON to a directory.

  Refer to "Start the JobSpy Daemon" for further details.

- -killd

  (optional) Terminates the JobSpy daemon.

- jobs

  (optional) Returns a list of current jobs with a variety of status information (e.g., job ID, current simulation time, start time, etc.).

- status

  (optional) Returns the status of the JobSpy daemon.

- <jobid> <command>

  (optional) Specifies a job ID to be processed by <command>. Use jobspy jobs to get a list of job IDs.

  <command> — A JobSpy simulator command. Refer to "Simulation Commands Available to JobSpy" for a list of valid commands.

Related Topics

- Monitoring Simulations with JobSpy
- Running the JobSpy GUI
- Simulation Commands Available to JobSpy
- Start the JobSpy Daemon

layout

This command allows you to perform a number of editing operations on custom GUI layouts, such as loading, saving, maximizing, and deleting.

The command options include:

- **layout active** – returns the current active window
- **layout current** – lists the current layout
- **layout delete** – removes the current layout from the *.modelsim* file (UNIX/Linux) or Registry (Windows)
- **layout load** – opens the specified layout
- **layout names** – lists all known layouts
- **layout normal** – minimizes the current maximized window
- **layout maximized** – return a 1 if the current layout is maximized, or a 0 if minimized
- layout restoretype — removes the list of window type(s) that will not be restored when a new layout is loaded.
- **layout save** – saves the current layout to the specified name
- layout showsuppresstypes — lists the window types that will not be restored when a new layout is loaded.
- layout suppresstype — adds the specified window type(s) to the list of types that will not be restored when a layout is reloaded.
- **layout togglezoom** – toggles the current zoom state of the active window (from minimized to maximized or maximized to minimized)
- **layout zoomactive** – maximizes the current active window
- **layout zoomwindow** – maximizes the specified window

Syntax

layout active

layout current

layout delete <name>

layout load <name>

layout names

layout normal

layout maximized

layout restoretype <window>

layout save <name>

layout showsuppresstypes

layout suppresstype <window>

layout togglezoom

layout zoomactive

layout zoomwindow <window>

Arguments

- <name>

  (required) Specifies the name of the layout.

- <window>

  (required) The window specification can be any format accepted by the view command. The window can be specified by its type (i.e., wave, list, objects, etc.), by the windowobj name (i.e., .main_pane.wave, .main_pain.library, etc.), or by the tab name (i.e., wave1, list3, etc.)

Related Topics

- Simulator GUI Layout Customization
- Configuring Default Windows for Restored Layouts

lecho

This command takes one or more Tcl lists as arguments and pretty-prints them to the Transcript window.

Syntax

lecho <args> ...

Arguments

- <args> ...

  Any Tcl list created by a command or user procedure. Specified as a space separated list.

Examples

- Print the Wave window configuration list to the Transcript window.

```
lecho [configure wave]
```

left

This command searches left (previous) for signal transitions or values in the specified Wave window.

It executes the search on signals currently selected in the window, starting at the time of the active cursor. The active cursor moves to the found location.

Use this command to move to consecutive transitions, to find the time at which a waveform takes on a particular value, or to find an expression of multiple signals that evaluates to true. See the right command for related functionality.

The procedure for using left entails three steps:

1. Click on the desired waveform.
2. Click on the desired starting location. (The seetime command can initially position the cursor from the command line, if desired.)
3. Issue the left command.

Returns: <number_found> <new_time> <new_delta>

Syntax

left [-expr {<expression>}] [-falling] [<n>] [-rising] [-value <sig_value> [-noglitch]] [-window <wname>]

Arguments

- -expr {<expression>}

  (optional) The waveform display is searched until the expression evaluates to a boolean true condition.

  <expression> — An expression that involves one or more objects, but limited to objects that have been logged in the referenced waveform display. An object may be specified either by its full path or by the shortcut label displayed in the Wave window.

  See GUI_expression_format for the format of the expression. The expression must be placed within braces ({}).

- -falling

  (optional) Searches for a falling edge on the specified signal if that signal is a scalar signal. If it is not a scalar signal, the option will be ignored.

- <n>

  (optional) Specifies to find the nth match where the default is 1. If less than n are found, the number found is returned with a warning message, and the cursor is positioned at the last match.

- -noglitch

  (optional) Looks at signal values only on the last delta of a time step. For use with the -value option only.

- -rising

  (optional) Searches for a rising edge on the specified signal if that signal is a scalar signal. If it is not a scalar signal, the option will be ignored.

- -value <sig_value>

  (optional) Specifies the value of the object to match.

  <sig_value> — A value specified in the same radix that the selected object is displayed. Case is ignored, but otherwise the value must be an exact string match. Don't-care bits are not supported. Only one signal can be selected, but that signal may be an array.

- -window <wname>

  (optional) Specifies an instance of the Wave window that is not the default. When <wname> is not specified, the default Wave window is used. Use the view command to change the default window.

  <wname> — The name of a Wave window not currently the default.

Examples

- Find the second time to the left at which the selected vector transitions to FF23, ignoring glitches.

  left -noglitch -value FF23 2

- Go to the previous transition on the selected signal.

  left

The following examples illustrate search expressions that use a variety of signal attributes, paths, array constants, and time variables. Such expressions follow the GUI_expression_format.

- Search left for an expression that evaluates to a boolean 1 when signal *clk* just changed from low to high and signal *mystate* is the enumeration reading and signal */top/u3/addr* is equal to the specified 32-bit hex constant; otherwise is 0.

  left -expr {clk'rising && (mystate == reading) && (/top/u3/addr == 32'habcd1234)}

- Search left for an expression that evaluates to a boolean 1 when the upper 8 bits of the 32-bit signal */top/u3/addr* equals hex ac.

  left -expr {(/top/u3/addr and 32'hff000000) == 32'hac000000}

- Search left for an expression that evaluates to a boolean 1 when logfile time is between 23 and 54 microseconds, clock just changed from low to high, and signal *mode* is enumeration writing.

  left -expr {((NOW > 23 us) && (NOW < 54 us)) && clk'rising && (mode == writing)}

Note

"Wave Window Mouse and Keyboard Shortcuts" are also available for next and previous edge searches. Tab searches right (next) and shift-tab searches left (previous).

Related Topics

- GUI_expression_format
- right
- seetime
- view

log

This command creates a wave log format (WLF) file containing simulation data for all HDL objects whose names match the provided specifications. Objects that are displayed using the add list and add wave commands are automatically recorded in the WLF file. By default the file is named *vsim.wlf* and stored in the current working directory. You can change the default name using the vsim -wlf option of the vsim command or by setting the WLFFilename variable in the *modelsim.ini* file.

If no port mode is specified, the WLF file contains data for all objects in the selected region whose names match the object name specification.

The WLF file contains a record of all data generated for the list and wave windows during a simulation run. Reloading the WLF file restores all objects and waveforms and their complete history from the start of the logged simulation run. See dataset open for more information.

For all transaction streams created through the SCV or Verilog APIs, logging is enabled by default. A transaction is logged to the WLF file if logging is enabled at the beginning of a simulation run when the design calls ::begin_transaction() or $begin_transaction. The effective start time of the transaction (the time passed by the design as a parameter to ::begin_transaction) is irrelevant. For example, a stream could have logging disabled between T1 and T2 and still record a transaction in that period, through retroactive logging after time T2. A transaction is always either entirely logged or entirely ignored.

Transaction streams created from an Questa Verification IP, as well as transaction instances, are not logged by default: they must be logged explicitly through the log, add wave, or add list commands. You can log Questa Verification IP completed transaction instances using -mvcreccomplete (all except add list), and transaction streams using -mvcall or -mvcovm.

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Note

The log command is also known as the "add log" command.

Syntax

log [-class <classtype>] [-howmany] [-filter <f> | -nofilter <f>]
{[-in] [-inout] [-out] | [-ports]} [-internal]
[-mvcall] [-mvcovm] [-mvcreccomplete] [-optcells]
[-recursive [-depth <level>]] **<object_name> …**

log -flush [<object>]

Arguments

- -class <classtype>

  (optional) Log all objects of a class specified in <classtype> that are generated after the command is invoked. Descends the hierarchy recursively to include all properties of <classtype> that are also classes.

<classtype> — The type of class to be logged.

Caution

Using the -class switch can result in a large amount of logged data.

- -depth <level>

  (optional) Restricts a recursive search (specified with the -recursive argument) to a certain level of hierarchy.

  <level> — Any non-negative integer. For example, if you specify -depth 1, the command descends only one level in the hierarchy.

- -filter <f> | -nofilter <f>

  (optional) Allows a one-time modification of the WildcardFilter in the command invocation. The add list command can take as many [-filter <f>] and [-nofilter <f>] arguments as you would like to specify. Valid filters, <f>, are exactly the same set of words that can be applied to the WildcardFilter. The filter(s) used during a command starts with the WildcardFilter and then applies the user specified filters, if any. The -filter values are added to the filter, the -nofilter values are removed from the filter. They are applied in the order specified so conflicts are resolved with the last specified wins.

- -flush [<object>]

  (optional) Forced the WLF file to write all buffered region and event data to the WLF file. By default, the region and event data is buffered and periodically written to the file, as appropriate. If <object> is specified, that object is first logged and then the file is flushed.

- -howmany

  (optional) Returns an integer indicating the number of signals found.

- -in

  (optional) Specifies that the WLF file is to include data for ports of mode IN whose names match the specification.

- -inout

  (optional) Specifies that the WLF file is to include data for ports of mode INOUT whose names match the specification.

- -internal

  (optional) Specifies that the WLF file is to include data for internal (non-port) objects whose names match the specification.

- -mvcall

  (optional) Specifies the inclusion of all Questa Verification IP protocol transactions when a wildcard is used (i.e. log -r /*). By default, Questa Verification IP transactions are sometimes excluded from logging with normal wildcard usage.

- -mvcovm

  (optional) Specifies the inclusion of all UVM and/or OVM sequence transactions when a wildcard is used (i.e. log -r /*). By default, UVM/OVM transactions are sometimes excluded from logging with normal wildcard usage.

- -mvcreccomplete

  (optional) Enables the logging of all Questa Verification IP transaction instances which have been recognized as completed, whether or not they become used as legal protocol. By default, only transactions that become used as legal protocol are visible, which can prevent the transaction instances of interest from being logged soon enough to observe an issue.

- -optcells

  (optional) Makes Verilog optimized cell ports visible when using wildcards. By default Verilog optimized cell ports are not selected even if they match the specified wildcard pattern.

- -out

  (optional) Specifies that the WLF file is to include data for ports of mode OUT whose names match the specification.

- -ports

  (optional) Specifies that the scope of the search is to include all ports, IN, INOUT, and OUT.

- -recursive

  (optional) Specifies that the scope of the search is to descend recursively into subregions. If omitted, the search is limited to the selected region. You can use the -depth argument to specify how far down the hierarchy to descend.

- <object_name>

  (required) Specifies the object name that you want to log. Must be specified as the final argument to the **log** command. Multiple object names are specified as a space separated list. Wildcard characters are allowed. Note that the *WildcardFilter* Tcl preference variable identifies types to ignore when matching objects with wildcard patterns.

  By default, wildcard card logging does not log the internals of cells. Refer to the +libcell | +nolibcell argument of the vlog command for more information.

Examples

- Log all objects in the design.

  log -r /*

- Log all output ports in the current design unit.

  log -out *

Related Topics

- add list
- add wave
- dataset alias
- dataset clear
- dataset close
- dataset config
- dataset config
- dataset info
- dataset list
- dataset open
- dataset restart
- dataset rename
- dataset save
- dataset snapshot
- nolog
- Recording Simulation Results With Datasets
- vlog +libcell | +nolibcell
- What is Questa Verification IP?
- Wildcard Characters
- WLFFilename variable

lshift

This command takes a Tcl list as an argument and shifts it in-place, one place to the left, eliminating the left-most element.

The number of shift places may also be specified. Returns nothing.

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

lshift <list> [<amount>]

Arguments

- <list>

  (required) Specifies the Tcl list to target with lshift. Must be specified as the first argument to the lshift command.

- <amount>

  (optional) Specifies the number of places to shift where the default is 1.

Examples

```
proc myfunc args {
   # throws away the first two arguments
   lshift args 2
   ...
   }
```

Related Topics

- Refer to the Tcl man pages (**Help > Tcl Man Pages**) for details.

lsublist

This command returns a sublist of the specified Tcl list that matches the specified Tcl glob pattern.

Arguments to this command are order dependent. Follow the order specified in the Syntax section.

Syntax

lsublist **<list>** <pattern>

Arguments

- <list>

  (required) Specifies the Tcl list to target with lsublist.

- <pattern>

  (required) Specifies the pattern to match within the <list> using Tcl glob-style matching.

Examples

- In the example below, variable ‘t’ returns "structure signals source".

```
set window_names "structure signals variables process source wave
list"
set t [lsublist $window_names s*]
```

Related Topics

- The **set** command is a Tcl command. Refer to the Tcl man pages (**Help > Tcl Man Pages**) for details.

mem compare

This command compares a selected memory to a reference memory or file. Must have the "diff" utility installed and visible in your search path in order to run this command.

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

mem compare {[-mem <ref_mem>] | [-file <ref_file>]} **[actual_mem]**

Arguments

- -mem <ref_mem>

  (optional) Specifies a reference memory to be compared with actual_mem.

  <ref_mem> — A memory record.

- -file <ref_file>

  (optional) Specifies a reference file to be compared with actual_mem.

  <ref_file> — A saved memory file.

- actual_mem

  (required) Specifies the name of the memory to be compared against the reference data. Must be specified as the final argument to the mem compare command.

mem display

This command prints to the Transcript window the memory contents of the specified instance. If the given instance path contains only a single array signal or variable, the signal or variable name need not be specified.

You can redirect the output of the mem display command into a file for later use with the mem load command. The output file can also be read by the Verilog $readmem system tasks if the memory module is a Verilog module and Verilog memory format (hex or binary) is specified.

Address radix, data radix, and address range for the output can also be specified, as well as special output formats.

By default, identical data lines are printed. To replace identical lines with a single line containing the asterisk character, you can enable compression with the -compress argument.

Note

The format settings are stored at the top of this file as a pseudo comment so that subsequent mem load commands can correctly interpret the data. Do not edit this data when manipulating a saved file.

Arguments to this command are order dependent. Please read through the argument descriptions for more information.

Syntax

mem display [-addressradix [d | h]] [-compress] [-dataradix <radix_type>] [-endaddress <end>][-format [bin | hex | mti]] [-noaddress] [-startaddress <st>] [-wordsperline <n>] [**<path>**]

Arguments

- -addressradix [d | h]

  (optional) Specifies the address radix for the default (MTI) formatted files.

  d — Decimal radix. (default if -format is specified as mti.)

  h — Hex radix.

- -compress

  (optional) Specifies that identical lines not be printed. Reduces the file size by replacing exact matches with a single line containing an asterisk. These compressed files are automatically expanded during a **mem load** operation.

- -dataradix <radix_type>

  (optional) Specifies the data radix for the default (MTI) formatted files. If unspecified, the global default radix is used.

  <radix_type> A specified radix type. Valid entries (or any unique abbreviations) are: binary, decimal, unsigned, octal, hex, symbolic, and default. If no radix is specified for an enumerated type, the symbolic representation is used.

You can change the default radix type for the current simulation using the radix command or make the default radix permanent by editing the DefaultRadix variable in the modelsim.ini file.

- -endaddress <end>

  (optional) Specifies the end address for a range of addresses to be displayed.

  <end> — Any valid address in the memory. If unspecified, the default is the end of the memory.

- -format [bin | hex | mti]

  (optional) Specifies the output format of the contents.

  bin— Specifies a binary output.

  hex— Specifies a hex output.

  mti — MTI format. (default).

- -noaddress

  (optional) Specifies that addresses not be printed.

- -startaddress <st>

  (optional) Specifies the start address for a range of addresses to be displayed.

  <st> — Any valid address in the memory. If unspecified, the default is the start of the memory.

- -wordsperline <n>

  (optional) Specifies how many words are to be printed on each line.

  <n> — Any positive integer where the default is an 80 column display width.

- <path>

  (required) Specifies the full path to the memory instance. The default is the current context, as shown in the Structure window. Indexes can be specified. Must be specified as the final argument to the mem display command.

Examples

- This command displays the memory contents of instance */top/c/mru_mem*, addresses 5 to 10:

  mem display -startaddress 5 -endaddress 10 /top/c/mru_mem

  returns:

  ```
  # 5: 110 110 110 110 110 000
  ```

- Display the memory contents of the same instance to the screen in hex format, as follows:

  mem display -format hex -startaddress 5 -endaddress 10 /top/c/mru_mem

  returns:

```
# 5: 6 6 6 6 6 0
```

Related Topics

- For details on MTI format, refer to the mem load description.

mem list

This command displays a flattened list of all memory instances in the current or specified context after a design has been elaborated.

Each instance line is prefixed by "VHDL:" or "Verilog:", depending on the type of model.

Returns the signal/variable name, address range, depth, and width of the memory.

Syntax

mem list [-r] [<path>]

Arguments

- -r

  (optional) Recursively descends into sub-modules when listing memories.

- <path>

  (optional) The hierarchical path to the location the search should start where the default is the current context, as shown in the Structure window.

Examples

- Recursively list all memories at the top level of the design.

  mem list -r /

  Returns:

  ```
  # Verilog: /top/m/mem[0:255](256d x 16w)
  #
  ```

- Recursively list all memories in */top2/uut*.

  mem list /top2/uut -r

  Returns:

  ```
  # Verilog: /top2/uut/mem[0:255] x 16w
  ```

mem load

This command updates the simulation memory contents of a specified instance. You can upload contents either from a memory data file, a memory pattern, or both. If both are specified, the pattern is applied only to memory locations not contained in the file.

A relocatable memory file is one that has been saved without address information. You can load a relocatable memory file into the instance of a memory core by specifying an address range on the mem load command line. If no address range (starting and ending address) is specified, the memory is loaded starting at the first location.

The order in which the data is placed into the memory depends on the format specified by the -format option. If you choose bin or hex format, the memory is filled low to high, to be compatible with $readmem commands. This is in contrast to the default MTI format, which fills the memory according to the memory declaration, from left index to right index.

For Verilog objects and VHDL integers and std_logic types: if the word width in a file is wider than the word width of the memory, the leftmost bits (msb) in the data words are ignored. To allow wide words use the -truncate option which will ignore the msb bits that exceed the memory word size. If the word width in the file is less than the width of the memory, and the leftmost digit of the file data is not 'X', then the leftmost bits are zero filled. Otherwise, they are X-filled.

The type of data required for the -filldata argument is dependent on the -filltype specified: a fixed value, or one that governs an incrementing, decrementing, or random sequence.

- For fixed pattern values, the fill pattern is repeatedly tiled to initialize the memory block specified. The pattern can contain multiple word values for this option.
- For incrementing or decrementing patterns, each memory word is treated as an unsigned quantity, and each successive memory location is filled in with a value one higher or lower than the previous value. The initial value must be specified.
- For a random pattern, a random data sequence will be generated to fill in the memory values. The data type in the sequence will match the type stored in the memory. For std_logic and associated types, unsigned integer sequences are generated. A seed value may be specified on the command line. For any given seed, the generated sequence is identical.

The interpretation of the pattern data is performed according to the default system radix setting. However, this can be overridden with a standard Verilog-style '<radix_char><data> specification.

Syntax

mem load {-infile <infile> | -filldata <data_word> [-infile <infile>]} [-endaddress <end>] [-fillradix <radix_type>] [-filltype {dec | inc | rand | value}] [-format [bin | hex | mti]] [<path>] [-skip <Nwords>] [-startaddress <st>] [-truncate]

Arguments

- -infile <infile>

  (required unless the -filldata argument is used) Updates memory data from the specified file.

  <infile> — The name of a memory file.

- -endaddress <end>

  (optional) Specifies the end address for a range of addresses to be loaded.

  <end> — Specified as any valid address in the memory.

- -filltype {dec | inc | rand | value}

  (optional, use with the -filldata argument) Fills in memory addresses in an algorithmic pattern starting with the data word specified in -filldata. If a fill pattern is used without a file option, the entire memory or specified address range is filled with the specified pattern. If both are specified, the pattern is applied only to memory locations not contained in the file.

  dec — Decrement each succeeding memory word by one digit.

  inc — Increment each succeeding memory word by one digit.

  rand — Randomly generate each succeeding memory word starting with the word specified by -filldata as the seed.

  value — Value (default) Substitute each memory word in the range with the value specified in -filldata.

- -filldata <data_word>

  (required unless -infile is used) Specifies a data word used to fill memory addresses in the pattern specified by -filltype.

  <data_word> — Specifies a data word. Must be in the same format as specified by the -fillradix switch.

- -fillradix <radix_type>

  (optional, use with -filldata) Specifies radix of the data specified by the -filldata switch.

  <radix_type> — Valid entries (or any unique abbreviations) are: binary, decimal, unsigned, octal, hex, symbolic, and default.

- -format [bin | hex | mti]

  (optional, use with -infile) Specifies the format of the file to be loaded.

  bin — Specifies binary data format.

  hex — Specifies hex format.

  mti — MTI format. (default).

  Specifies the format of the file to be loaded. The bin and hex values are the standard Verilog hex and binary memory pattern file formats. These can be used with Verilog memories, and with VHDL memories composed of std_logic types.

In the MTI memory data file format, internal file address and data radix settings are stored within the file itself. Thus, there is no need to specify these settings on the mem load command line. If a format specified on the command line and the format signature stored internally within the file do not agree, the file cannot be loaded.

- <path>

  (optional) The hierarchical path to the memory instance. If the memory instance name is unique, shorthand instance names can be used. The default is the current context, as shown in the Structure window.

  Memory address indexes can be specified in the instance name also. If addresses are specified both in the instance name and the file, only the intersection of the two address ranges is populated with memory data.

- -skip <Nwords>

  (optional) Specifies the number of words to be skipped between each fill pattern value. Used with -filltype and -filldata.

  <Nwords> — Specified as an unsigned integer.

- -startaddress <st>

  (optional) Specifies the start address for a range of addresses to be loaded.

  <st> — Any valid address in the memory.

- -truncate

  (optional) Ignores any most significant bits (msb) in a memory word which exceed the memory word size. By default, when memory word size is exceeded, an error results.

Examples

- Load the memory pattern from the file *vals.mem* to the memory instance */top/m/mem*, filling the rest of the memory with the fixed-value 1‘b0.

  mem load -infile vals.mem -format bin -filltype value -filldata 1'b0 /top/m/mem

  When you enter the **mem display** command on memory addresses 0 through 12, you see the following:

```
mem display -startaddress 0 -endaddress 12 /top/m/mem
#  0: 0000000000000000 0000000000000001 0000000000000010 0000000000000011
#  4: 0000000000000100 0000000000000101 0000000000000110 0000000000000111
#  8: 0000000000001000 0000000000001001 0000000000000000 0000000000000000
# 12: 0000000000000000
```

- Load the memory pattern from the file *vals.mem* to the memory instance */top/m/mru_mem*, filling the rest of the memory with the fixed-value 16'Hbeef.

  mem load -infile vals.mem -format hex -st 0 -end 12 -filltype value -filldata 16'Hbeef /top/m/mru_mem

- Load memory instance */top/mem2* with two words of memory data using the Verilog Hex format, skipping 3 words after each fill pattern sequence.

  mem load -filltype value -filldata "16'hab 16'hcd" /top/mem2 -skip 3

- Load memory instance */top/mem* with zeros (0).

  mem load -filldata 0 /top/mem

- Truncate the msb bits that exceed the maximum word size (specified in HDL code).

  mem load -format h -truncate -infile data_files/data.out /top/m_reg_inc/mem

Related Topics

- mem save

mem save

This command saves the contents of a memory instance to a file in any of the supported formats: Verilog binary, Verilog hex, and MTI memory pattern data.

This command works identically to the mem display command, except that its output is written to a file rather than a display.

The order in which the data is placed into the saved file depends on the format specified by the -format argument. If you choose bin or hex format, the file is populated from low to high, to be compatible with $readmem commands. This is in contrast to the default mti format, which populates the file according to the memory declaration, from left index to right index.

You can use the mem save command to generate relocatable memory data files. The -noaddress option omits the address information from the memory data file. You can later load the generated memory data file using the memory load command.

Syntax

mem save **-outfile <filename>** [-addressradix {dec | hex}] [-dataradix <radix_type>]
[-format {bin | hex | mti}] [-compress | -noaddress] [<path>]
[-startaddress <st> -endaddress <end>] [-wordsperline <Nwords>]

Arguments

- -outfile <filename>

  (required) Specifies that the memory contents are to be stored in a file.

  <filename> — The name of the file where the specified memory contents are to be stored.

- -addressradix {dec | hex}

  (optional) Specifies the address radix for the default mti formatted files.

  dec — Decimal (default).

  hex — Hexadecimal.

- -compress

  (optional) Specifies that only unique lines are printed, identical lines are not printed. Mutually exclusive with the -noaddress switch.

- -dataradix <radix_type>

  (optional) Specifies the data radix for the default mti formatted files.

  <radix_type> — Valid entries (or any unique abbreviations) are: binary, decimal, unsigned, octal, hex, and symbolic.

  You can change the default radix for the current simulation using the radix command. You can change the default radix permanently by editing the DefaultRadix variable in the modelsim.ini file.

- -endaddress <end>

  (optional) Specifies the end address for a range of addresses to be saved.

  <end> — Any valid address in the memory.

- -format {bin | hex | mti}

  (optional) Specifies the format of the output file.

  bin— Binary data format.

  hex— Hexadecimal format.

  mti — MTI format. (default).

  The bin and hex values are the standard Verilog hex and binary memory pattern file formats. These can be used with Verilog memories, and with VHDL memories composed of std_logic types.

  In the MTI memory data file format, internal file address and data radix settings are stored within the file itself.

- -noaddress

  (optional) Prevents addresses from being printed. Mutually exclusive with the -compress switch.

- <path>

  (optional) The hierarchical path to the location of the memory instance. The default is the current context, as shown in the Structure window.

- -startaddress <st>

  (optional) Specifies the start address for a range of addresses to be saved.

  <st> — Any valid address in the memory.

- -wordsperline <Nwords>

  (optional) Specifies how many memory values are to be printed on each line.

  <Nwords> — Any unsigned integer where the default assumes an 80 character display width.

Examples

- Save the memory contents of the instance */top/m/mem(0:10)* to *memfile*, written in the mti radix.

  mem save -format mti -outfile memfile -start 0 -end 10 /top/m/mem

  The contents of *memfile* are as follows:

```
// memory data file (do not edit the following line - required for mem
load use)
// format=mti addressradix=d dataradix=s version = 1.0
0: 0000000000000000 0000000000000001 0000000000000010 0000000000000011
4: 0000000000000100 0000000000000101 0000000000000110 0000000000000111
8: 0000000000001000 0000000000001001 xxxxxxxxxxxxxxxx
```

Related Topics

- mem display
- mem load

mem search

This command finds and prints to the screen the first occurring match of a specified memory pattern in the specified memory instance. Shorthand instance names are accepted.

Optionally, you can instruct the command to print all occurrences. The search pattern can be one word or a sequence of words.

Syntax

mem search {**-glob <word> [<word>…]** | **-regexp <word> [<word>…]**}
[-addressradix {dec | hex}] [-dataradix <radix_type>] [-all] [-replace <word> [<word>…]]
[-startaddress <address>] [-endaddress <address>] [<path>]

Arguments

- -glob <word> [<word>…]

  (required unless using -regexp) Specifies the value of the pattern, accepting glob pattern syntax for the search.

  <word> — Any word pattern. Multiple word patterns are specified as a space separated list. Wildcards are accepted in the pattern.

  This argument and -regexp are mutually exclusive arguments.

- -regexp <word> [<word>…]

  (required unless using -glob) Specifies the value of the pattern, accepting regular expression syntax for the search.

  <word> — Any word pattern. Wildcards are accepted in the pattern. Multiple word patterns are specified as a space separated list.

  This argument and -glob are mutually exclusive arguments.

- -addressradix {dec | hex}

  (optional) Specifies the radix for the address being displayed.

  dec — Decimal (default).

  hex — Hexadecimal.

- -all

  (optional) Searches the specified memory range and returns all matching occurrences to the transcript. By default only the first matching occurrence is printed.

- -dataradix <radix_type>

  (optional) Specifies the radix for the memory data being displayed.

  <radix_type> — Can be specified as symbolic, binary, octal, decimal, unsigned, or hex. By default the radix displayed is the system default.

You can change the default radix for the current simulation using the radix command. You can change the default radix permanently by editing the DefaultRadix variable in the modelsim.ini file.

- -endaddress <address>

  (optional) Specifies the end address for a range of addresses to search.

  <address> — Any valid address in the memory.

- <path>

  (optional) Specifies the hierarchical path to the location of the memory instance. The default is the current context, as shown in the Structure window.

- -replace <word> [<word>…]

  (optional) Replaces the found patterns with a designated pattern.

  <word> — A word pattern Multiple word patterns are accepted, separated by a single space. No wildcards are allowed in the replaced pattern.

- -startaddress <address>

  (optional) Specifies the start address for a range of addresses to search.

  <address> — Any valid address in the memory.

Examples

- Search for and print to the screen all occurrences of the pattern 16‘Hbeef in */uut/u0/mem3*:

  mem search -glob 16‘Hbeef -dataradix hex /uut/u0/mem3

  Returns:

  ```
  #7845: beef
  #7846: beef
  #100223: beef
  ```

- Search for and print only the first occurrence of 16‘Hbeef in the address range 7845:150000, replacing it with 16‘Hcafe in /uut/u1/*mem3*:

  mem search -glob 16‘Hbeef -d hex -replace 16‘Hcafe -st 7846 -end 150000 /uut/u1/mem3

  Returns:

  ```
  #7846: cafe
  ```

- Replace all occurrences of 16‘Hbeef with 16‘Habe in */uut/u1/mem3*:

  mem search -glob 16‘Hbeef -r 16‘Habe -addressadix hex -all /uut/u1/mem3

  Returns:

```
#1ea5: 2750
#1ea6: 2750
#1877f: 2750
```

- Search for and print the first occurrence any pattern ending in f:

  mem search -glob "*f"

- Search for and print the first occurrence of this multiple word pattern:

  mem search -glob "abe cafe" /uut/u1/mem3

- Search for patterns "0000 0000" or "0001 0000" in *m/mem*:

  mem search -data hex -regexp {000[0|1] 0{4}} m/mem -all

- Search for a pattern that has any number of 0s followed by any number of 1s as a memory location, and which has a memory location containing digits as the value:

  mem search -regexp {^0+1+$ \d+} m/mem -all

- Search for any initialized location in a VHDL memory:

  mem search -regexp {[^U]} -all <vhdl_memory>

next

This command continues a search after you have invoked the search command.

Syntax

next <window_name> [-window <wname>]

Arguments

- <window_name>

  (required) Specifies one window in which to continue searching. Can be one of the following windows: Signals, Objects, Variables, Locals, Source, List, Wave, Process, Structure. Unique abbreviations are accepted.

- -window <wname>

  (optional) Specifies an instance of the window that is not the default.

  <wname> — The name of a window instance that is not the default. Otherwise, the default window is used.

  Use the view command to change the default window.

Related Topics

- **search**

noforce

This command removes the effect of any active force commands on the selected HDL objects. and also causes the object's value to be re-evaluated.

You can use noforce on signals within SystemC modules, with the following limitations:

- Only mixed language boundaries types are supported.
- Individual bits and slices may not be forced or unforced.

Syntax

noforce **<object_name> …**

Arguments

- <object_name>

  (required) Specifies the name of an object. Must match an object name used in a previous force command. Multiple object names may be specified as a space separated list. Wildcard characters are allowed.

Related Topics

- force
- Wildcard Characters

nolog

This command suspends writing of data to the wave log format (WLF) file for the specified signals.

A flag is written into the WLF file for each signal turned off, and the GUI displays "-No Data-" for the signal(s) until logging (for the signal(s)) is turned back on. Logging can be turned back on by issuing another log command or by doing a nolog -reset.

Because use of the nolog command adds new information to the WLF file, WLF files created when using the nolog command cannot be read by older versions of the simulator.

Transactions written in SCV or Verilog are logged automatically, and can be removed with the nolog command. A transaction is logged into the .wlf file if logging is enabled (in other words, if no nolog command has disabled it) for that stream at the time when the transaction was begun. An entire span of a transaction is either logged or not logged, regardless of the begin and end times specified for that transaction.

Syntax

nolog [-all] [-depth <level>] [-howmany] [-in] [-inout] [-internal] [-out] [-ports] [-recursive] [-reset] [<object_name>…]

Arguments

- -all

  (optional) Turns off logging for all signals currently logged.

- -depth <level>

  (optional) Restricts a recursive search (specified with the -recursive argument) to a certain level of hierarchy.

  <level> — An integer greater than or equal to zero. For example, if you specify -depth 1, the command descends only one level in the hierarchy.

- -howmany

  (optional) Returns an integer indicating the number of signals found.

- -in

  (optional) Turns off logging only for ports of mode IN whose names match the specification.

- -inout

  (optional) Turns off logging only for ports of mode INOUT whose names match the specification.

- -internal

  (optional) Turns off logging only for internal (non-port) objects whose names match the specification.

- -out

  (optional) Turns off logging only for ports of mode OUT whose names match the specification.

- -ports

  (optional) Specifies that the scope of the search is to include all ports.

- -recursive

  (optional) Specifies that the scope of the search is to descend recursively into subregions. If omitted, the search is limited to the selected region. You can use the -depth argument to specify how many levels of the hierarchy to descend.

- -reset

  (optional) Turns logging back on for all unlogged signals.

- <object_name>…

  (optional) Specifies the object name which you want to unlog. Multiple object names may be specified as a space separated list. Wildcard characters are allowed.

Examples

- Unlog all objects in the design.

  nolog -r /*

- Turn logging back on for all unlogged signals.

  nolog -reset

Related Topics

- add list
- add wave
- log

notepad

This command opens a simple text editor. It may be used to view and edit ASCII files or create new files.

This mode can be changed from the Notepad Edit menu.

Returns nothing.

Syntax

notepad [<filename>] [-r | -edit]

Arguments

- <filename>

  (optional) Name of the file to be displayed.

- -r

  (optional) Specifies read-only mode.

- -edit

  (optional) Specifies editing mode. Will not save changes to an existing file that has the Read-only attribute turned on. (default)

noview

This command closes a window in the Questa SIM GUI. To open a window, use the view command.

Syntax

noview <window_name>…

Arguments

- <window_name>…

  (required) Specifies the window(s) to close. Multiple window types may be specified in a space separated list. Wildcards permitted. At least one type (or wildcard) is required.

  Refer to the view command for a complete list of possible arguments.

  You can also close Source windows using the tab or file name.

Examples

- Close the Wave window named "wave1".

  noview wave1

- Close all List windows.

  noview List

Related Topics

- view

nowhen

This command deactivates selected when commands.

Syntax

nowhen [<label>]

Arguments

- <label>

  (optional) Specifies an individual when command. Wildcards may be used to select more than one when command.

Examples

- Deactivate the when command labeled 99.

  nowhen 99

- Deactivate all when commands.

  nowhen *

Related Topics

- when

onbreak

This command is used within a DO file and specifies one or more scripts to be executed when running a script that encounters a breakpoint in the source code.

The onbreak setting will affect any run commands that follow the onbreak statement until another onbreak command is issued. If a DO file is executed from within the script, the DO file script will inherit the onbreak setting specified prior to execution unless and until another onbreak command is given, in which case, that onbreak setting will be in effect until the DO file script completes at which point execution will return to the calling script and the calling script's onbreak setting is restored. For more information, refer to Breakpoint Flow Control in Nested DO files. The script must be followed by a run command to take effect.

Default behavior when there is no onbreak setting in effect is definced by the OnBreakDefaultAction Tcl variable. If the OnBreakDefaultAction variable is not defined, the simulator default is to resume execution.

Use an empty string to change the onbreak command back to the default behavior:

```
onbreak ""
```

In this case, the script will resume after a breakpoint occurs (after any associated bp command string is executed).

If you specify this command in a DO file with an empty script (" "), the default behavior is to resume execution of the DO file. The onbreak command with no argument will return the current user specified script or empty if the default onbreak is in effect.

Syntax

onbreak <script>; <script>...

Arguments

- <script>

  (optional) Any command or script can be used as an argument to onbreak. If you want to use more than one command or script, use a semicolon to separate them or place them on multiple lines and enclose the entire script in braces ({}) or quotation marks (“ “). You must use the onbreak command before a run, run -continue, or step command. If a run or step comand is issued within an onbreak script, the script will return immediately and any following commands will not be executed. It is an error to execute any commands within an onbreak command string following any of the run commands. This restriction applies to any macros or Tcl procedures used in the onbreak command string.

Examples

- Examine the value of the HDL object data when a breakpoint is encountered. Then continue the run command.

  ```
  onbreak {exa data ; cont}
  ```

- Resume execution of the DO file on encountering a breakpoint.

onbreak resume

- This set of commands test for assertions. Assertions are treated as breakpoints if the severity level is greater than or equal to the current BreakOnAssertion variable setting (refer to modelsim.ini Variables). By default a severity level of failure or above causes a breakpoint; a severity level of error or below does not.

```
set broken 0
onbreak {
  lassign [runStatus -full] status fullstat
  if {$status eq "error"} {
    # Unexpected error, report info and force an error exit
    echo "Error: $fullstat"
    set broken 1
    resume
  } elseif {$status eq "break"} {
    # If this is a user break, then
    # issue a prompt to give interactive
    # control to the user
    if {[string match "user_*" $fullstat]} {
      pause
    } else {
      # Assertion or other break condition
      set broken 2
      resume
    }
  } else {
    resume
  }
}
run -all
if {$broken == 1} {
  # Unexpected condition.  Exit with bad status.
  echo "failure"
  quit -force -code 3
} elseif {$broken == 2} {
  # Assertion or other break condition
  echo "error"
  quit -force -code 1
} else {
  echo "success!"
}
quit -force
```

Related Topics

- abort
- bp
- do
- onerror
- resume
- status
- Useful Commands for Handling Breakpoints and Errors
- DO Files

onElabError

This command specifies one or more commands to be executed when an error is encountered during the elaboration portion of a vsim command. The command is used by placing it within a DO file script.

Use the onElabError command without arguments to return to a prompt.

Syntax

onElabError {[<command> [; <command>] …]}

Arguments

- <command>

  (optional) Any command can be used as an argument to onElabError. If you want to use more than one command, use a semicolon to separate the commands, or place them on multiple lines. The entire command string must be placed in braces ({}).

Related Topics

- do
- vsim

onerror

This command is used within a DO file script before a run command; it specifies one or more commands to be executed when a running script encounters an error.

Using the onerror command without arguments will return the current onerror command string. Use an empty string (onerror “”) to change the onerror command back to its default behavior. Use onerror with a resume command to allow an error message to be printed without halting the execution of the DO file.

You can also set the global OnErrorDefaultAction Tcl variable to dictate what action Questa SIM takes when an error occurs. To set the variable on a permanent basis, you must define the variable in a *modelsim.tcl* file (Refer to “The modelsim.tcl File” for details).

When your onerror command is successful, the DO file script will continue normally, unless your command instructs the tool to quit, for example:

```
onerror {quit -f}
```

or

```
onerror {break}
```

However, if your **onerror** command is not successful, the simulator will be halted, for example:

```
onerrror {add wave b}
```

when you don’t have a signal named b.

The onerror command is executed when a Tcl command (for example, break.) encounters an error in the DO file that contains the onerror command (note that a run command does not necessarily need to be in process). Conversely, OnErrorDefaultAction will run even if the DO file does not contain a local onerror command. This can be useful when you run a series of DO files from one script, and you want the same behavior across all DO files.

Syntax

onerror {[<command> [; <command>] …]}

Arguments

- <command>

  (optional) Any command can be used as an argument to onerror. If you want to use more than one command, use a semicolon to separate the commands, or place them on multiple lines. The entire command string must be placed in braces ({}).

Examples

- Force the simulator to quit if an error is encountered while the DO file is running.

  ```
  onerror {quit -f}
  ```

Related Topics

- abort
- do
- onbreak
- resume
- status
- Useful Commands for Handling Breakpoints and Errors
- DO Files

onfinish

This command controls simulator behavior when encountering $finish or sc_stop() in the design code. When you specify this command without an argument, it returns the current setting.

Syntax

onfinish [ask | exit | final | stop | default]

Arguments

- ask

  (optional) In batch mode, the simulation will exit; in GUI mode, the user is prompted for action.

- exit

  (optional) The simulation exits without asking for any confirmation.

- final

  (optional) The simulation executes all finish blocks before exiting.

- stop

  (optional) The simulation ends but remains loaded in memory, allowing for easier post-simulation tasks.

- default

  (optional) Uses the current setting for the OnFinish variable in the *modelsim.ini* file.

Related Topics

- OnFinish modelsim.ini variable
- sc_stop Behavior
- vsim -onfinish

pa autotestplan

This command creates a test plan file based on coverage information in your power aware simulation.

Refer the the section "Generating and Using a Test Plan Based on Power Aware Coverage" in the *Power Aware Simulatin User's Manual* for more information.

Syntax

pa autotestplan [-format={xml | ucdb}] [-filename=<filename>]

Arguments

- -format={xml | ucdb}

  Specifies the filetype format to write. The default is UCDB format.

  If you generate the test plan in XML format, you can use the xml2ucdb command to convert to UCDB format.

- -filename=<filename>

  Specifies the filename of the test plan. The default is *QuestaPowerAwareTestplan.ucdb*.

pa msg

This command controls the display of warning and error messages that occur during Power Aware dynamic checks during simulation run time. You can run this command at the VSIM prompt or in a do file. Refer to the section “Dynamic Power Aware Check Message Control” for more detailed usage information.

Syntax

pa msg [-all] [-domain <string>] [-enable | -disable] [-glitch_window <real><time_spec>] [<message_num> …] [-pa_checks=<check_value>] [-ports <port_list>] [-scope <string>] [-severity {note | warning | error | fatal}] [-stopafter <integer>] [-strategy <string>] []

Arguments

- -all

  (optional) Specifies that this command applies to all messages from dynamic power aware checks. The default behavior is to apply the command to all messages other than those with a severity of NOTE.

- -domain <string>

  (optional) Restricts command to output messages pertaining only to the power domain specified by <string>. This will not have any impact on messages related to checks not associated with any power domain, such as those controlled with -pa_checks arguments uml and umi.

- -enable | -disable

  (optional) Specifies whether to display or suppress messages provided by Power Aware dynamic checking during simulation run time (default severity of warning/error). By default, all messages are displayed (-enable).

- -glitch_window <real><time_spec>

  Specifies the interval for glitch checking on control lines. Note that there is no space between the two arguments: <real> and <time_spec>. This option applies only to glitch checks, as specified with -pa_checks=ugc.

  <real> — any positive real number or zero (0). The default value is zero if you do not specify this switch.

  <time_spec> — specification for seconds (s) or fractions of a second (fs, ps, ns, us, ms). The default value is the specification for your simulation.

- <message_num>

  Specifies the identification number of one or more messages you want to suppress, display, or set the severity for (use spaces to separate multiple message numbers). Optional. If you do not provide a message number, -enable and -disable affect all dynamic checking messages.

- -pa_checks=<check_value>

  Applies the pa msg command to Power Aware dynamic checks, using the same argument values available for the vopt -pa_checks=<check_value> command.

- -ports <port_list>

  Restricts messages to those pertaining to the ports listed in <port_list>. This switch applies only to the following -pa_checks arguments: irc, ifc, icp, and it.

  <port_list> — Grouping of space-separated port names enclosed in braces ({}).

- -scope <string>

  Restricts command to output messages pertaining only to the scope specified by <string>. The default behavior is to show all messages from the top scope of the design. This option is only meaningful when specified with -domain, -strategy, or -ports.

  <string> — Hierarchical representation of the scope. If this value begins with a slash (/) the string is read as is, such as -scope /tb/TOP/mid1. If this value does not begin with a slash (/) the string is prepended with the top level scope of the design, where -scope TOP/mid1 would be read as “/tb/TOP/mid1”.

- -severity {note | warning | error | fatal}

  Sets the severity type for one or more messages (specified by message_id).

- -stopafter <integer>

  Stops the simulation after one or more messages are displayed the specified number of times.

- -strategy <string>

  Restricts command to output messages pertaining only to the strategy specified by <string>. This will not have any impact on messages related to checks not associated with any strategy, such as those controllwed with -pa_checks arguments p, t, uml, umi, and npu.

Examples

- Suppress messages 8908 and 8903 from simulation time 0 to time 50, then allow those messages to be displayed from time 50 to time 100. (By default, all other messages are displayed from time 0 to time 100.)

```
pa msg -disable 8908 8903
run 50
pa msg -enable 8908 8903
run 50
```

- Suppress all messages from simulation time 125 to time 275, then allow all messages to be displayed after time 275.

```
run 125
pa msg -disable
run 150
pa msg -enable
run 350
```

- Set the severity of message number 8901 to “warning.”

```
pa msg -severity warning 8901
```

- Enable message 8905, if it was disabled previously, and change its severity to “fatal” (which stops the simulation on first occurrence).

```
pa msg -enable -severity fatal 8905
```

- Restrict messages to all objects of power domain PD_mid1 in the scope /tb/dut/mid1:

  pa msg -scope /tb/dut/mid1 -domain PD_mid1

- Restrict messages to isolation checks belonging to strategy iso_mid1:

```
pa msg -pa_checks=i -strategy iso_mid1
```

- Stop the simulation if a retention, toggle, or always-on check occurs three times.

```
pa msg -stopafter 3 -pa_checks=r+t+a
```

- Disable all isolation functionality and clamp value check for isolation strategy "iso_mid" belonging to power domain "/tb/TOP/mid1/PD_mid"

```
pa msg -disable -pa_checks=ifc+icp -scope /tb/TOP/mid1 -domain
PD_mid -strategy iso_mid
```

Related Topics

- “Dynamic Power Aware Check Message Control”

pa report

This command is the final step of the power aware report generation flow, which writes out all of the power aware reports to the current working directory. Refer to the section “Generating Reports for Power Aware” in the *Power Aware Simulation User’s Manual* for more information.

Prerequisites for this command include:

- Run vopt with the -pa_genrpt option to collect the necessary information.
- (optional) Run vopt with the -pa_reportdir option to change the default location of the reports.

Syntax

pa report

Arguments

none

Related Topics

- “Generating Reports for Power Aware”

pause

This command interrupts the execution of a macro and allows you to perform interactive debugging of a macro file. The command is placed within the macro to be debugged.

Syntax

pause

Arguments

None.

Description

When a macro is interrupted during execution, the macro returns the prompt:

```
VSIM(paused)>
```

This “pause” prompt notifies you that a macro has been interrupted.

When a macro is paused, you can invoke another macro. If the second macro is interrupted, you can continue invoking macros up to a nesting level of 50 macros.

The status command lists summary information about all interrupted macros.

Use the resume command to resume execution of the macro. Use the abort command to stop execution of some or all of the macros.

Related Topics

- abort
- do
- resume
- run
- status

pop

This command moves the specified number of call frames up the C callstack.

This command is used with C Debug.

Syntax

pop <#_of_levels>

Arguments

- <#_of_levels>

  (optional) Specifies the number of call frames to move up the C callstack. If unspecified, movement is up one level.

Examples

- Move up 1 call frame.

  pop

- Move up 4 call frames.

  pop 4

Related Topics

- push
- C Debug

power add

This command specifies the signals or nets to monitor for power information. When power add is called on a signal or net, vsim keeps a record of any toggling activity of that signal. The information is sent a file when you run the power report command. This data can be translated and used by third-party power analysis tools.

The basic steps for using this command are:

1. Add the signals or nets of interest with the power add command.
2. Run the simulation with the run command.
3. Produce a report with the power report command.

Note

You can use the power off command to disable monitoring between runs and then use the power on command to resume monitoring.

Arguments to this command are order-dependent. Please read the argument descriptions for more information.

Syntax

power add [[-in] [-out] [-inout] | [-ports]] [-internal] [-nocellnet] [-r] <signals_nets> ...

Arguments

- -in

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode IN if they match the <signal_net> specification.

- -out

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode OUT if they match the <signal_net> specification.

- -inout

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode INOUT if they match the <signal_net> specification.

- -ports

  (optional) *For use with wildcard searches.* Specifies that the scope of the listing is to include ports of modes IN, OUT, or INOUT if they match the <signal_net> specification.

- -internal

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include internal objects (non-port objects) if they match the <signal_net> specification.

- -nocellnet

  (optional) Prevents the vsim command from monitoring cell-net for toggling or any power-related activity.

- -r

  (optional) Searches recursively on a wildcard specified for the signal or net.

- <signals_nets> ...

  (required) Specifies the signal or net path to monitor. Must be specified as the final argument to the power add command.

  You can specify multiple names and also use wildcards. The signals and nets must refer to:

  - VHDL — records, arrays of records, bit, bit_vector, signed and unsigned, arrays of signed and unsigned, std_logic, integer/natural/positive, and one- and two-dimensional arrays.

    Refer to the power add -vhdlint argument for information about VHDL integers.

  - Verilog — nets, and one- and two-dimensional arrays (packed and unpacked).

  When using wildcards, the -in, -inout, -internal, -out, and -ports arguments filter the qualifying signals.

- -vhdlint

  (optional) adds VHDL integers to a bsaif dump from the power report command using the -bsaif option.

If you specify more than one of the arguments (-in, -inout, -internal, -out, or -ports), the logical OR of the arguments is performed.

Related Topics

- power off
- power on
- power report
- power reset

power off

The power off command works in conjunction with the power add command to make vsim stop updating toggle activity data for the specified signal or net.

After this command is executed, every subsequent run command ignores the power add command for the specified arguments to the power off command.

Arguments to this command are order-dependent. Please read the argument descriptions for more information.

Syntax

power off [-all] [[-in] [-out] [-inout] | [-ports]] [-internal] [-r] **<signals_nets> ...**

Arguments

- -all

  (optional) For use with wildcard searches. Specifies that the scope of the search is to include inputs, inouts, and outputs. if they match the <signal_nets> specification.

- -in

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode IN if they match the <signal_nets> specification.

- -out

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode OUT if they match the <signal_nets> specification.

- -inout

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode INOUT if they match the <signal_nets> specification.

- -ports

  (optional) *For use with wildcard searches.* Specifies that the scope of the listing is to include ports of modes IN, OUT, or INOUT if they match the <signal_nets> specification.

- -internal

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include internal objects (non-port objects) if they match the <signal_nets> specification.

- -r

  (optional) Searches recursively on a wildcard specified for the signal or net.

- <signals_nets> ...

  (required) Specifies the signal or net to monitor. Must be specified as the final argument to the power off command.

  You can specify multiple names and also use wildcards. The signals and nets must refer to:

- VHDL — records, arrays of records, bit, bit_vector, signed, unsigned, arrays of signed and unsigned, std_logic, integer/natural/positive, and one- and two-dimensional arrays

  Refer to the power add -vhdlint argument for information about VHDL integers.

- Verilog — nets, and one- and two-dimensional arrays (packed and unpacked).

When using wildcards, the -in, -inout, -internal, -out, and -ports arguments filter the qualifying signals.

If you specify more than one of the arguments (-in, -inout, -internal, -out, or -ports), the logical OR of the arguments is performed.

Examples

Assume that signal /top/a toggles every 5ns.

- Without running the power off command:

  power add /top/a
  run 400ns
  power report

  Returns:

```
Node      Tc      Ti     Time At 1    Time At 0    Time At X
-----------------------------------------------------------
/top/a    80       0          200          200            0
-----------------------------------------------------------
```

- Running the power off command (and resuming with power on):

  power add /top/a
  run 100ns
  power off
  run 100ns
  power on
  run 200ns
  power report

  Returns:

```
Node      Tc      Ti     Time At 1    Time At 0    Time At X
-----------------------------------------------------------
/top/a    60       0          150          150            0
-----------------------------------------------------------
```

Related Topics

- power add
- power on
- power report
- power reset

power on

This command works in conjunction with the power add command to make vsim begin or resume updating toggle activity data for the specified signal or net.

After this command is executed, every subsequent run command implements the power add command.

Arguments to this command are order-dependent. Please read the argument descriptions for more information.

Syntax

power on [-all] [[-in] [-out] [-inout] | [-ports]] [-internal] [-r] **<signals_nets> ...**

Arguments

- -all

  (optional) For use with wildcard searches. Specifies that the scope of the search is to include inputs, inouts, and outputs. if they match the <signal_nets> specification.

- -in

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode IN if they match the <signal_nets> specification.

- -out

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode OUT if they match the <signal_nets> specification.

- -inout

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode INOUT if they match the <signal_nets> specification.

- -ports

  (optional) *For use with wildcard searches.* Specifies that the scope of the listing is to include ports of modes IN, OUT, or INOUT if they match the <signal_nets> specification.

- -internal

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include internal objects (non-port objects) if they match the <signal_nets> specification.

- -r

  (optional) Searches recursively on a wildcard specified for the signal or net.

- <signals_nets> ...

  (required) Specifies the signal or net to monitor. Must be specified as the final argument to the power on command.

You can specify multiple names and also use wildcards. The signals and nets must refer to:

- VHDL — records, arrays of records, bit, bit_vector, signed, unsigned, arrays of signed and unsigned, std_logic, integer/natural/positive, and one- and two-dimensional arrays

  Refer to the power add -vhdlint argument for information about VHDL integers.

- Verilog — nets, and one- and two-dimensional arrays (packed and unpacked).

When using wildcards, the -in, -inout, -internal, -out, and -ports arguments filter the qualifying signals.

If you specify more than one of the arguments (-in, -inout, -internal, -out, or -ports), the logical OR of the arguments is performed.

Examples

Assume that signal /top/a toggles every 5ns.

- Without running the power off command:

  power add /top/a
  run 400ns
  power report

  Returns:

```
Node      Tc      Ti     Time At 1    Time At 0    Time At X
-----------------------------------------------------------
/top/a    80       0          200          200            0
-----------------------------------------------------------
```

- Running the power off command (and resuming with power on):

  power add /top/a
  run 100ns
  power off
  run 100ns
  power on
  run 200ns
  power report

  Returns:

```
Node      Tc      Ti     Time At 1    Time At 0    Time At X
-----------------------------------------------------------
/top/a    60       0          150          150            0
-----------------------------------------------------------
```

Related Topics

- power add
- power off
- power report
- power reset

power report

This command reports power information for the objects specified with the power add command.

The report can be in either a tabular ASCII format (-file) or a Switching Activity Interchange Format (SAIF) format (-bsaif).

Data produced by these commands can be translated and used by third-party tools used for power analysis. The power report command is intended to be used as follows:

1. Add the objects of interest with the power add command.
2. Run the simulation with the run command.
3. Produce the report with the power report command.

Syntax

power report [-all] [-noheader] [-file <filename>] [-bsaif <filename>]

Arguments

- -all

  (optional) Writes information on all objects logged with power add.

  If you do not specify this argument, the report lists only those signals or nets that have a toggle count that is non-zero.

- -noheader

  (optional) Suppresses the header to aid in post processing.

  This argument has no effect on the output from the -bsaif switch.

- -file <filename>

  (optional) Specifies an ASCII-format power report is to be created.

  <filename> — A user-specified name for the report.

  If you do not specify this argument or the -bsaif argument, the tabular ASCII-format power report is returned to the transcript. You can specify both -file and -bsaif on the same command line, resulting in both reports being generated.

- -bsaif <filename>

  (optional) Specifies a backward-SAIF format power report.

  <filename> — A user-specified name for the report.

  If you do not specify this argument or the -file argument, the tabular ASCII-format power report is returned to the transcript. You can specify both -file and -bsaif on the same command line, resulting in both reports being generated.

Description

The report format for each line is:

```
Node, Tc, Ti, Time at 1, Time at 0, Time at X
```

- Node — The hierarchical path of the signal, net or port.
- Tc — (toggle count) The number of 0->1 and 1->0 transitions.
- Ti — (hazard count) The number of 0/1->X, and X->0/1 transitions.

Note

Note that if a signal is initialized at X, and later transitions to 0 or 1, it is not counted as a hazard.

- Time at X — The length of time spent at each of the three respective states.

Examples

Example Reports

The following example is from the -file output of the tabular ASCII-format power report.

```
#
# Power Report Interval
# 1100000 ps
#
# Power Report    Node     Tc     Ti    Time At 1    Time At 0    Time At X
# ---------------------------------------------------------------------------
# /test_sm/out_wire(7)      1      0    669000 ps    420000 ps     11000 ps
# /test_sm/out_wire(6)      2      0    520000 ps    569000 ps     11000 ps
# /test_sm/out_wire(5)      3      0    149000 ps    940000 ps     11000 ps
# /test_sm/out_wire(4)      2      0     60000 ps   1029000 ps     11000 ps
# /test_sm/out_wire(3)      1      0    669000 ps    420000 ps     11000 ps
# ...
```

The following example is from the -bsaif output of the backward SAIF format power report. This file contains Header information (between SAIFILE and DURATION entries) and specific instance information (INSTANCE entries)

```
(SAIFILE
(SAIFVERSION "2.0")
(DIRECTION "backward")
(DESIGN )
(VENDOR "Mentor Graphics")
(PROGRAM_NAME "vsim")
(VERSION "2.3b")
(TIMESCALE 100ps)
(DIVIDER /)
(DURATION 1200000 ps)
(INSTANCE
   (INSTANCE test_sm
      (NET out_wire(7)
         (T0 420000 ps) (T1 769000 ps) (TX 11000 ps)
         (TC 1) (IG 0)
      )
```

```
      )
   )
   (INSTANCE
      (INSTANCE test_sm
         (NET out_wire(6)
            (T0 660000 ps) (T1 529000 ps) (TX 11000 ps)
            (TC 3) (IG 0)
         )
      )
   )

   ...
```

Related Topics

- power add
- power off
- power on
- power reset

power reset

This command selectively resets power information to zero for the signals or nets specified with the power add command.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

power reset [-all] [[-in] [-out] [-inout] | [-ports]] [-internal] [-r] **<signals_nets> ...**

Arguments

- -all

  (optional) For use with wildcard searches. Specifies that the scope of the search is to include inputs, inouts, and outputs. if they match the <signal_nets> specification.

- -in

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode IN if they match the <signal_nets> specification.

- -out

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode OUT if they match the <signal_nets> specification.

- -inout

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include ports of mode INOUT if they match the <signal_nets> specification.

- -ports

  (optional) *For use with wildcard searches.* Specifies that the scope of the listing is to include ports of modes IN, OUT, or INOUT if they match the <signal_nets> specification.

- -internal

  (optional) *For use with wildcard searches.* Specifies that the scope of the search is to include internal objects (non-port objects) if they match the <signal_nets> specification.

- -r

  (optional) Searches recursively on a wildcard specified for the signal or net.

- <signals_nets> ...

  (required) Specifies the signal or net to monitor. Must be specified as the final argument to the power reset command.

  You can specify multiple names and also use wildcards. The signals and nets must refer to:

  - VHDL — records, arrays of records, bit, bit_vector, signed, unsigned, arrays of signed and unsigned, std_logic, integer/natural/positive, and one- and two-dimensional arrays

Refer to the power add -vhdlint argument for information about VHDL integers.

- Verilog — nets, and one- and two-dimensional arrays (packed and unpacked).

When using wildcards, the -in, -inout, -internal, -out, and -ports arguments filter the qualifying signals.

If you specify more than one of the arguments (-in, -inout, -internal, -out, or -ports), the logical OR of the arguments is performed.

Related Topics

- power add
- power off
- power on
- power report

precision

This command determines how real numbers display in the graphic interface (e.g., Objects, Wave, Locals, and List windows). It does not affect the internal representation of a real number and therefore precision values over 17 are not allowed.

Executing the precision command without any arguments returns the current precision setting.

Syntax

precision [<digits>[#]]

Arguments

- <digits>[#]

  (optional) Specifies the number of digits to display where the default is 6.

  # — A suffix that forces the display of trailing zeros. See examples for more details.

Examples

- Results in 4 digits of precision.

  precision 4

  For example:

  ```
  1.234 or 6543
  ```

- Results in 8 digits of precision including trailing zeros.

  precision 8#

  For example:

  ```
  1.2345600 or 6543.2100
  ```

- Results in 8 digits of precision but doesn’t print trailing zeros.

  precision 8

  For example:

  ```
  1.23456 or 6543.21
  ```

printenv

This command prints to the Transcript window the current names and values of all environment variables.

If variable names are given as arguments, returns only the names and values of the specified variables.

Syntax

printenv [<var>…]

Arguments

- <var>…

  (optional) Specifies the name(s) of the environment variable(s) to print.

Examples

- Print all environment variable names and their current values.

  printenv

  Returns:

  ```
  # CC = gcc
  # DISPLAY = srl:0.0
  …
  ```

- Print the specified environment variables:

  printenv USER HOME

  Returns:

  ```
  # USER = vince
  # HOME = /scratch/srl/vince
  ```

process report

This command creates a textual report of all processes displayed in the Process Window.

Syntax

process report [-file <filename>] [-append]

Arguments

- -file <filename>

  (optional) Creates an external file where raw process data will be saved. If -file is not specified, then the output is redirected to stdout.

  <filename> — A user-specified name for the file.

- -append

  (optional) Specifies that process data is to be appended to the current process report file. If this option is not used, the process data will overwrite the existing process report file.

profile clear

This command clears any performance data that has been gathered during previous executions of the run command. Use the profile on command to begin profiling.

After this command is executed, all profiling data is reset.

This command has no effect on the current profiling session. The last profile on or profile off command will still be in effect.

Prerequisites

First enable profiling with the profile on command.

Syntax

profile clear

Arguments

None

Related Topics

- profile interval
- profile off
- profile on
- profile open
- profile option
- profile reload
- profile report
- profile save
- profile summary
- Profiling Performance and Memory Use

profile interval

This command selects the frequency with which the profiler collects samples during a run command.

Prerequisites

First enable profiling with the profile on command.

Syntax

profile interval [<sample_frequency>]

Arguments

- <sample_frequency>

  Any integer from 1 to 999 that represents how many milliseconds (ms) to wait between each sample collected during a profiled simulation run where the default is 10 ms.

  If the profile interval command is executed without arguments, the current sample frequency is returned.

Related Topics

- profile clear
- profile off
- profile on
- profile open
- profile option
- profile reload
- profile report
- profile save
- profile summary
- Profiling Performance and Memory Use

profile off

This command disables runtime memory allocation and statistical performance profiling.

Syntax

profile off [-assertions] [-classes] [-cvg] [-qdas] [-solver] [-m] [-p]

Arguments

- -assertions

  (optional) Disables fine-grain analysis of memory capacity data being collected for assertions and cover directives.

- -classes

  (optional) Disables fine-grain analysis of memory capacity data being collected for class objects.

- -cvg

  (optional) Disables fine-grain analysis of memory capacity data being collected for covergroups.

- -qdas

  (optional) Disables fine-grain analysis of memory capacity data being collected for queues, dynamic arrays, associative arrays.

- -solver

  (optional) Disables fine-grain analysis of memory capacity data being collected for randomize () calls.

- -m

  (optional) Disables memory allocation profiling only.

- -p

  (optional) Disables statistical performance profiling only.

Related Topics

- profile clear
- profile interval
- profile on
- profile open
- profile option
- profile reload
- profile report
- profile save
- profile summary
- Profiling Performance and Memory Use

profile on

This command enables runtime memory allocation and statistical performance profiling. Deletes any existing profile data if that data came from a previous invocation of the profile open command.

After this command is executed, every subsequent run command will be profiled.

Note

It is suggested that you not run the memory allocation and statistical sampling profilers at the same time. The analysis of the memory allocation can skew the results of the statistical sampling profiler.

Syntax

profile on [-assertions] [-classes] [-cvg] [-qdas] [-solver] [-p]
[-m [-file <filename> | -fileonly <filename>]]

Arguments

- -assertions

  (optional) Enables fine-grain analysis of memory capacity data collection for assertions and cover directives.

- -classes

  (optional) Enables fine-grain analysis of memory capacity data collection for class objects.

- -cvg

  (optional) Enables fine-grain analysis of memory capacity data collection for covergroups.

- -qdas

  (optional) Enables fine-grain analysis of memory capacity data collection for queues, dynamic arrays, associative arrays.

- -solver

  (optional) Enables fine-grain analysis of memory capacity data collection for randomize () calls.

- -m

  (optional) Enables memory allocation profiling only.

- -p

  (optional) Enables statistical performance profiling only.

- -file <filename>

  (optional) Allows creation of a raw profile data file that can be post-processed later using the “profile reload” command. Saves memory profile data into both an external file and internal data structures.

- -fileonly <filename>

  (optional) Allows creation of a raw profile data file that can be post-processed later using the “profile reload” command. Saves memory profile data into an external file only, not to internal data structures.

Examples

- The following set of commands enables the profiler, runs the simulation for 1000 nanoseconds, and outputs the profiling data to *perf.rpt*.

```
profile on
run 1000 ns
profile report -file perf.rpt
```

Related Topics

- profile clear
- profile interval
- profile off
- profile open
- profile option
- profile reload
- profile report
- profile save
- profile summary
- Profiling Performance and Memory Use

profile open

This command opens a database containing profile data created with the profile save command. It deletes any existing profile data, whether collected during previous activity in simulation mode or from a previous invocation of the profile open command.

This command is useful for analyzing profile data from another user.

Syntax

profile open **<filename>**

Arguments

- <filename>

  (required) An absolute or relative pathname to a file containing the profile database.

Related Topics

- profile clear
- profile interval
- profile off
- profile on
- profile option
- profile reload
- profile report
- profile save
- profile summary
- Profiling Performance and Memory Use

profile option

This command changes how profiling data are reported. The command also acts like a toggle: invoking it the first time turns on the option; invoking it a second time turns the option off. After each execution of the command the new setting is returned to the transcript.

Prerequisites

First enable profiling with the **profile on** command.

Syntax

profile option collapse_sections [on | off | status]

profile option collect_calltrees [on | off | status]

Arguments

- collapse_sections [on | off | status]

  (required) Groups profiling data by section. A section consists of regions of code such as VHDL processes, functions, or Verilog *always* blocks. By default all profiling data are reported on a per line basis.

  on — (optional) Enables profiling of data by section. (default)

  off — (optional) Disables profiling of data by section

  status — (optional) Returns the current setting of the profile option collapse_sections command.

  If executed without options, the profile option collapse_sections command acts as a toggle.

- collect_calltrees [on | off | status]

  (required) Collects data for call trees, showing which functions or routines call which others. By default this information is not collected. Simulation time and resource usage will increase if you enable collection of this data.

  on — (optional) Enables profiling of data by section.

  off — (optional) Disables profiling of data by section. (default)

  status — (optional) Returns the current setting of the profile option collect_calltrees command.

Examples

- Enable profiling of collapsed processes and functions.

  profile option collapse_sections on

  Returns:

  ```
  # Profiling will now report collapsed processes and functions
  ```

- Turn off reporting of collapsed processes and functions.

profile option collapse_sections

Returns:

```
# Profiling will now NOT report collapsed processes and functions
```

Related Topics

- profile clear
- profile interval
- profile off
- profile on
- profile open
- profile reload
- profile report
- profile save
- profile summary
- Profiling Performance and Memory Use

profile reload

This command reads in raw profile data from an external file created during memory allocation profiling. The profile report command and the Profile and Profile Details windows of the user interface can be used to view the data. The intent of the raw profile files is to allow analysis of memory profile data in cases where the memory required for the design plus the memory required for internal profiling data exceeds the memory capacity of the machine.

The profile reload command clears all performance and memory profiling data collected up to that point. Any currently loaded design is terminated (similar to the quit -sim command), and run-time profiling is turned off. If a new design is loaded after you have read the raw profile data, then all internal profile data is cleared, but run-time profiling is not turned back on.

Prerequisites

You must first execute profile on -m -file <filename> or profile on -m -fileonly <filename>.

Syntax

profile reload <filename>

Arguments

- <filename>

  (required) Designates the name of the external file where raw profile data will be saved.

Related Topics

- profile clear
- profile interval
- profile off
- profile on
- profile open
- profile option
- profile report
- profile save
- profile summary
- Profiling Performance and Memory Use

profile report

This command outputs profiling data that have been gathered up to the point that you execute the command.

Prerequisites

First enable profiling using the profile on command or the vsim -memprof command.

Syntax

profile report

[-callercallee <func> |
-calltree |
-cutoff <percentage> |
-du [<du_name>] [-showcalls] |
-file <filename> |
-functoinst <func> |
-instofdef <inst> [-inclusiveDuMatch 0 |1] |
-m |
-onexit |
-p |
-ranked |
-structural [-level <positive_integer>] [<rootname>] [-showcalls]]

[-assertions] | [-classes] | [cvg] | [-qdas] | [-solver]

Arguments

- -assertions

  (optional) Reports memory usage data for SystemVerilog assertions and cover directives.

- -calltree

  (optional) Reports a hierarchical callstack list of statistical performance and memory allocation data. (default)

- -callercallee <func>

  (optional) Creates a ranked report of all callers and callees of the specified function.

  <func> — A function name (for Systemc, PLI, FLI) or a <.v/.vhd-filename>:<line#>

- -classes

  (optional) Reports memory usage data for the current number of objects allocated, the current memory allocated for class object, the peak memory allocated and peak time.

- -cutoff <percentage>

  (optional) Filters out entries in the report that had less than <percentage> of time spent in them where the default is to report all entries (i.e., 0%) when -cutoff is not specified.

  <percentage> — (required) Any non-negative integer 0 - 100.

- cvg

  (optional) Reports memory usage data for the number of covergroups, cross, bins and memory allocated.

- -du [<du_name>] [-showcalls]

  (optional) Reports a list of statistical performance and memory allocation data organized by design unit. Optional.

  <du_name> — (optional) Reports information about a specific design unit only. If omitted, the report includes all design units.

  -showcalls — (optional) Lists function callstacks beneath each design unit. If omitted, functional callstacks are not shown in the report.

- -file <filename>

  (optional) Specifies that report data be saved to a file. Default is to write the report to the Transcript window.

  <filename> — Any valid filename. May include special characters and numbers.

- -functoinst <func>

  (optional) Creates a ranked profile report of all instances of the specified function.

  <func> — A function name (for Systemc, PLI, FLI) or a <.v/.vhd-filename>:<line#>

- -instofdef <inst>

  (optional) Creates a ranked report of all instances with the same definition as the specified instance, showing profile results for each.

  <inst> — The hierarchical pathname of the specified instance.

- -inclusiveDuMatch 0 |1

  (optional) Determines how strict the instance definition is for the -instofdef <inst> argument.

  0 — Includes in the report only instances that reference the exact design unit (for example, a specific entity/architecture pair).

  1 — (default) Includes all instances that reference design units with the same primary name. For example if your design has multiple architectures for a VHDL entity, a value of 1 will cause matching for all instances that use the same entity.

- -m

  (optional) Displays memory allocation data in the report. If -m is not specified, the profile report will include memory allocation data if the memory profiler was previously enabled and memory information was collected during a run.

- -onexit

  (optional) Causes the command to be executed when the simulator exits. Allows you to queue multiple profile report commands.

- -p

  (optional) Displays statistical performance samples in the report. If -p is not specified, the profile report will include performance statistics if the performance profiler was previously enabled and profile samples were collected during a run.

- -qdas

  (optional) Reports memory usage data for queues, dynamic arrays, and associative arrays.

- -ranked

  (optional) Reports a ranked list of statistical performance and memory allocation data.

- -solver

  (optional) Reports memory usage data for calls to randomize() and memory usage.

- -structural [-level <positive_integer>] [<rootname>] [-showcalls]

  (optional) Reports a structural list of statistical performance and memory allocation data.

  -level <positive_integer> — (optional) Determines how far to expand instance hierarchy. If omitted, the report includes all levels.

  <positive_integer> — Any positive integer.

  <rootname> — (optional) Causes the report to be rooted at the specified instance. If not specified, the report contains all roots and any orphan samples.

  -showcalls — (optional) Lists function callstacks beneath each instance. If omitted, functional callstacks are not shown in the report.

Examples

- This set of commands enables the statistical sampling profiler, runs the simulation for 1000 nanoseconds, and outputs the calltree profiling data to a file named *perf.rpt*.

```
profile on
run 1000 ns
profile report -file perf.rpt
```

- Output ranked profile data for instances accounting for greater than 2% of the simulation time.

```
profile report -ranked -cutoff 2
```

- Output to file *perf.rpt* ranked profile data for all instances that use the same entity as does instance */top/c/s0*.

```
profile report -file perf.rpt -instofdef /top/c/s0
```

Related Topics

- profile clear
- profile interval
- profile off
- profile on
- profile open
- profile option
- profile reload
- profile save
- profile summary
- Profiling Performance and Memory Use

profile save

This command saves the profile data to an external database when Questa SIM shuts down. This command is useful for sending profile information to another user for analysis.

Prerequisites

Profiling must be active when this command is invoked except when specifying -onexit. Use the profile on command to begin profiling.

Syntax

profile save {[-onexit] <filename> | -onexitstatus}

Arguments

- <filename>

  (required) An absolute or relative path and filename to save the profile database.

- -onexit

  (optional) Causes the command to be executed when the simulator exits. You can queue multiple profile save commands by using this option.

- -onexitstatus

  (optional) Reports queued profile save -onexit <filename> commands.

Description

This command allows you to save profile information to more than one profile database (the <filename> argument) and to report queued profile save commands using the -onexitstatus option.

Use the profile open command to gain access to the database information.

Examples

The following sequence of profile save commands shows the difference between immediate profile save commands and those queued by using the command's -onexit option.

```
profile on
profile interval 2
run 24 ms
profile save -onexitstatus
profile save xyz.pdb
profile save -onexitstatus
profile save -onexit abc.pdb
profile save -onexitstatus
# profile save abc.pdb
profile save -onexit def.pdb
profile save -onexitstatus
# profile save def.pdb
# profile save abc.pdb
```

Related Topics

- profile clear
- profile interval
- profile off
- profile on
- profile open
- profile option
- profile reload
- profile report
- profile summary
- Profiling Performance and Memory Use

profile summary

This command outputs profiling data that is broken into categories based on context in which the callstacks are found.

Prerequisites

First enable profiling using the **profile on** command or the vsim -autoprofile command.

Syntax

profile summary [-file <filename>] [-header] [-noheader] [-tcl]

Arguments

- -file <filename>

  (optional) Directs the output to a file.

  <filename> — A user-specified name. Can include numbers and special characters.

- -header

  (optional) Includes header data in the summary information. Refer to Table 2-5 for more information.

- -noheader

  (optional) Prevents header data from being written with the summary information. Refer to Table 2-5 for more information.

- -tcl

  (optional) Output returned as Tcl list.

Note

Headers are printed to the output depending on which arguments are specified. Refer to Table 2-5 for more information.

Description

The profile summary command accounts for each callstack entry by incrementing the count in the applicable category listed:

VHDL — Samples in VHDL files

Verilog — Samples in Verilog files.

user_c — PLI code

syslib — shared system libraries like clib, part of the operating system. This would include functions like strcmp (string compare), memcpy (memory copy), and so on.

native — Samples there were found in VHDL/Verilog source code.

solver — Samples from the solver.

assertion — Samples from assertion code.

log — Samples associated with logging data.

kernel — Samples that didn't fall into any other category.

Each callstack is inspected, starting at the leaf and working up toward the root. The tests for different categories are made in the order shown above. As soon as a match is made, that category is used for that callstack. A callstack entry may apply to more than one category, it is accumulated into the first matching category. This is why there are VHDL and Verilog categories separate from the 'native' category. The VHDL/Verilog categories are used if the first match is from code that can be literally traced to a line in an HDL file.

The 'native' category is used for HDL code that doesn't match to a particular HDL source file, but is identified as being in code that came from your HDL.

Headers are returned depending on the arguments specified with profile summary.

Table 2-5. profile summary Header Rules for Return

profile summary Arguments	Default	Arguments Specified with -header	Arguments Specified with -noheader
profile summary	no	yes	no
profile summary -tcl	no	yes	no
profile summary -file <filename>	yes	yes	no
profile summary -tcl -file <filename>	no	yes	no

Examples

profile summary

```
# Profile Summary:
#                kernel    154   43.26%
#                user_c    117   32.87%
#         SystemVerilog     55   15.45%
#                native     30    8.43%
#       ===============  =====  =======
#                 Total    356  100.00%
```

Related Topics

- profile clear
- profile interval
- profile off
- profile on
- profile open
- profile option
- profile reload
- profile report
- profile save
- Profiling Performance and Memory Use

project

This command is used to perform common operations on projects.

Prerequisites

Some arguments to this command require a project to be opened with either the project new or project open command. Some argument must be used outside of a simulation session. Please read the argument descriptions for more information.

Syntax

project [addfile <filename> [<file_type>] [<folder_name>]] | [addfolder <foldername> [<folder_parent>]] | [calculateorder] | [close] | [compileall [-n]] | [compileorder] | [compileoutofdate [-n]] | [delete <filename>] | [filenames] | [env] | [history] | [new <home_dir> <proj_name> [<defaultlibrary>] [<intialini>] [0 | 1]] | [open <project>] | [removefile <filename>]

Arguments

- addfile <filename> [<file_type>] [<folder_name>]

  (optional) Adds the specified file to the current project. Requires a project to be open.

  <filename> — (required) The name of an existing file.

  <file_type> — (optional) The HDL file type of the file being added. For example do for a *.do* file.

  <folder_name> — (optional) Places the file in an existing folder created with project addfolder command. If no folder name is specified the file will be placed in the top level folder.

- addfolder <foldername> [<folder_parent>]

  (optional) Creates a project folder within the project. Requires a project to be open.

  <foldername> — (required) Any string.

  <folder_parent> — (optional) Places <foldername> in an existing parent folder. If <folder_parent> is unspecified, <foldername> is placed at the top level.

- calculateorder

  (optional) Determines the compile order for the project by compiling each file, then moving any compiles that fail to the end of the list. This is repeated until there are no more compile errors.

- close

  (optional) Closes the current project.

- compileall [-n]

  (optional) Compiles all files in the project using the defined compile order.

  -n — (optional) Returns a list of the compile commands this command would execute, without actually executing the compiles.

- compileorder

  (optional) Returns the current compile order list.

- compileoutofdate [-n]

  (optional) Compiles all files that have a newer date/time stamp than the last time the file was compiled.

  -n — Returns a list of the compile commands this command would execute, without actually executing the compiles.

- delete <filename>

  (optional) Deletes a project file.

  <filename> — Any *.mpf* file.

- filenames

  Returns the absolute pathnames of all files contained in the currently open project.

- env

  (optional) Returns the current project file and path.

- history

  (optional) Lists a history of manipulated projects. Must be used outside of a simulation session.

- new <home_dir> <proj_name> [<defaultlibrary>] [<intialini>] [0 | 1]

  (optional) Creates a new project under a specified home directory with a specified name and optionally a default library. The name of the work library will default to "work" unless specified. A new project cannot be created while a project is currently open or a simulation is in progress.

  <home_dir> — The path to the new project directory within the current working directory.

  <proj_name> — Specifies a name for the new project. The file will be saved as an *.mpf* file

  <defaultlibrary> — Specifies a name for the default library.

  <intialini> — An optional *modelsim.ini* file can be specified as a seed for the project file by using the initialini option. If initialini is an empty string, then Questa SIM uses the current *modelsim.ini* file when creating the project. You must specify a default library if you want to specify initialini.

  0 — (default) Copies all library mappings from the specified <initialini> file into the new project.

  1 — Copies library mappings referenced in an "others" clause in the initial *.ini* file.

- open <project>

  (optional) Closes any currently opened project and opens a specified project file (must be a valid *.mpf* file), making it the current project. Changes the current working directory to the project's directory. Must be used outside of a simulation session.

- removefile <filename>

  (optional) Removes the specified file from the current project.

Examples

- Make */user/george/design/test3/test3.mpf* the current project and changes the current working directory to */user/george/design/test3*.

  project open /user/george/design/test3/test3.mpf

- Execute current project library build scripts.

  project compileall

property list

This command changes one or more properties of the specified signal, net, or register in the List window.

The properties correspond to those you can set by selecting **View > Signal Properties** (List window). At least one argument must be used.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

property list [-window <wname>] [-label <label>] [-radix <type>]
[-radixenumnumeric | -radixenumsymbolic] [-trigger 0 | 1] [-width <number>] **<pattern>**

Arguments

- -window <wname>

  (optional) Specifies a particular List window when multiple instances of the window exist (e.g., list2). If no window is specified the default window is used; the default window is determined by the most recent invocation of the view command and has " - Default" appended to the name.

  <wname> — Specifies a List window other than the default list window.

- -label <label>

  (optional) Specifies a label to appear at the top of the List window column identifying <pattern>.

  <label> — A user-specified name. Can include numbers and special characters.

- -radix <type>

  (optional) Specifies the radix for List window objects.

  <type> — Any valid radix type: binary, ascii, unsigned, decimal, octal, hex, sfixed, symbolic, time, ufixed, and default. If no radix is specified for an enumerated type, the default representation is used.

  You can change the default radix for the current simulation using the radix command. You can change the default radix permanently by editing the DefaultRadix variable in the *modelsim.ini* file.

  If you specify a radix for an array of a VHDL enumerated type, Questa SIM converts each signal value to 1, 0, Z, or X.

- -radixenumnumeric

  (optional) Causes Verilog and SystemC enums to be displayed as numbers (formatted by the current radix). This overrides the default behavior of always showing enums symbolically.

- -radixenumsymbolic

  (optional) Restores the default behavior of displaying Verilog and SystemC enums as symbols by reversing the action of the -radixenumnumeric option.

- -trigger 0 | 1

  (optional) Toggles whether changes to <pattern> add data in the list window.

  0 — Does not write changes to <pattern> to the list window.

  1 — Writes changes to <pattern> to the list window.

- -width <number>

  (optional) Specifies the desired column width for the objects matching the specified pattern.

  <number> — Any positive integer 1 through 256.

- <pattern>

  (required) Specifies a name or wildcard pattern to match the full pathnames of the signals, nets, or registers for which you are defining the property change. Must be specified as the final argument to the property list command.

property wave

This command changes one or more properties of the specified signal, net, or register in the Wave window.

The properties correspond to those you can set by selecting **View > Signal Properties** (Wave window). At least one argument must be used.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

property wave [-window <wname>] [-color <color>] [-format {analog | literal | logic}] [-height <number>] [-offset <number>] [-radix <type>] [-radixenumsymbolic] [-scale <n>] **<pattern>**

Arguments

- -window <wname>

  (optional) Specifies a particular Wave window when multiple instances of the window exist. If no window is specified the default window is used; the default window is determined by the most recent invocation of the view command and has " - Default" appended to the name.

  <wname> — Specifies the instance of the Wave window to use (for example, wave2).

- -color <color>

  (optional) Specifies a different color is to be used for the waveform.

  <color> — The color value may be a color name or its hex value.

- -format {analog | literal | logic}

  (optional) Specifies the format of the wave form in <pattern>.

  analog — (optional) Displays a waveform whose height and position is determined by the -scale and -offset arguments.

  literal — (optional) Displays the waveform as a box containing the object value (if the value fits the space available).

  logic — (optional) Displays values as 0, 1, X, or Z.

- -height <number>

  (optional) Specifies the height (in pixels) of the waveform.

  <number> — Any integer.

- -offset <number>

  (optional) Specifies the waveform position offset in pixels. Valid only when -format is specified as analog.

  <number> — Any non-negative integer.

- -radix <type>

  (optional) Specifies the radix for Wave window objects.

  <type> — Any valid radix type (or unique abbreviation): binary, ascii, unsigned, decimal, octal, hex, symbolic, time, and default. If no radix is specified for an enumerated type, the default representation is used.

  You can change the default radix for the current simulation using the radix command. You can change the default radix permanently by editing the DefaultRadix variable in the *modelsim.ini* file.

  If you specify a radix for an array of a VHDL enumerated type, Questa SIM converts each signal value to 1, 0, Z, or X.

- -radixenumnumeric

  (optional) Causes Verilog and SystemC enums to be displayed as numbers (formatted by the current radix). This overrides the default behavior of always showing enums symbolically.

- -radixenumsymbolic

  (optional) Restores the default behavior of displaying Verilog and SystemC enums as symbols by reversing the action of the -radixenumnumeric option.

- -scale <n>

  (optional) Specifies the waveform scale relative to the unscaled size value of 1. Valid only when -format is specified as analog.

  <n> — Any non-negative integer.

- <pattern>

  (required) Specifies a name or wildcard pattern to match the full path names of the signals, nets, or registers for which you are defining the property change. Must be specified as the final argument to the property wave command

push

This command moves the specified number of call frames down the C callstack.

This command is used with C Debug. Refer to "C Debug" for more information.

Syntax

push <#_of_levels>

Arguments

- <#_of_levels>

  (optional) Specifies the number of call frames to move down the C callstack. If unspecified, 1 level is assumed.

Examples

- Move down 1 call frame.

  push

- Move down 4 call frames.

  push 4

Related Topics

- pop
- C Debug

pwd

This Tcl command displays the current directory path in the Transcript window.

Syntax

pwd

Arguments

None

questasim

This command starts the QuestaSim GUI without prompting you to load a design.

This command is valid only for Windows platforms and may be invoked in one of three ways:

- from the DOS prompt
- from a QuestaSim shortcut
- from the Windows Start > Run menu

To use questasim arguments with a shortcut, add them to the target line of the properties of that shortcut. (As expected, arguments also work on the DOS command line.)

The simulator may be invoked from either the QuestaSim> prompt after the GUI starts or from a DO file called by questasim.

Syntax

questasim [-do <macrofile>] [<license_option>]

Arguments

- -do <macrofile>

  (optional) Executes a specified DO file path when questasim is invoked.

  <macrofile> — Any valid DO file path, for example:

- <license_option>

  (optional) Refer to the vsim [<license_option>] argument.

Note

In addition to the macro called by this argument, if a DO file is specified by the STARTUP variable in *modelsim.ini*, it will be called when vsim is invoked.

Examples

- Start QuestaSim and execute the DO file *run.do*.

  Start > Run

  questasim -do <install_dir>\examples\tutorials\verilog\run1.do

Related Topics

- vsim
- do
- Using a Startup File

quietly

This command turns off transcript echoing for the specified command.

Syntax

quietly <command>

Arguments

- <command>

  (required) Specifies the command for which to disable transcript echoing. Any results normally echoed by the specified command will not be written to the Transcript window. To disable echoing for all commands use the transcript command with the -quietly option.

Related Topics

- transcript

quit

This command exits the simulator.

If you want to stop the simulation using a when command, you must use a stop command within your when statement, you must not use an exit or a quit command. The stop command acts like a breakpoint at the time it is evaluated.

Syntax

quit [-f | -force] [-sim] [-code <integer>]

Arguments

- -f | -force

  (optional) Quits without asking for confirmation. If omitted, Questa SIM asks you for confirmation before exiting. (The -f and -force arguments are equivalent.)

- -sim

  (optional) Unloads the current design in the simulator without exiting Questa SIM. All files opened by the simulation will be closed including the WLF file (*vsim.wlf*).

- -code <integer>

  (optional) Quits the simulation and issues an exit code.

  <integer> — This is the value of the exit code. You should not specify an exit code that already exists in Questa SIM. Refer to the section "Exit Codes" in the User's Manual for a list of existing exit codes. You can also specify a variable in place of <integer>.

  You should always print a message before running the quit -code command to explicitly state the reason for exiting.

Examples

Refer to the Examples section of the exit command for an example of using the -code argument. The quit and exit commands behave similarly in this regard.

qverilog

The qverilog command compiles (vlog), optimizes (vopt), and simulates (vsim) Verilog and SystemVerilog designs in a single step. It combines the compile, elaborate, and simulate phases together, as some users may be accustomed to doing with simulation tools from other vendors. This command is provided to ease these users' transition to Questa SIM.

The qverilog command invokes vlog, vopt, and then vsim. All standard vlog (and vopt) arguments are supported and are applied directly to the qverilog command line. All vsim options are supported and are applied through the qverilog -R argument.

All compiled but unreferenced modules become top level instances. Modules which are compiled in files included by vlog -y, but are unreferenced, also become top level modules unless -svext=-uslt is specified with the qverilog command.

You can directly enter either C or C++ file onto the qverilog command line. Questa SIM automatically processes them using the SystemVerilog Direct Programming Interface (DPI). Refer to "DPI and the vlog Command" for details. If your design contains DPI export tasks or functions, it is recommended that you use the vlog/vsim flow.

By default, qverilog runs the simulation and quits automatically by invoking an implicit "run -all; quit -f". However, if you invoke qverilog with -do, -gui, or -i, qverilog invokes the simulator and keeps it open until you explicitly quit Questa SIM.

The qverilog command creates a work library named work in the current directory, if not present already.

The command arguments listed below are only those unique to the qverilog command. This command also supports all vlog command arguments.

Syntax

qverilog [<vlog_and_vopt_options>] [-ccflags "opts"] [-do <dofile>] [-gui] [-i]
[-l <logfile>] <filename> [-ldflags "opts"]
[-R <vsim_options>] [-stats [=[+ | -]<feature>[,[+ | -]<mode>]] [-work <library_name>]

Arguments

- [<vlog_and_vopt_options>]

  All vlog and vopt options are supported. For example, if you are running qverilog on a SystemVerilog design, you need to add the -sv argument to the command line.

- -ccflags "opts"

  Specifies all C/C++ compiler options. Options are in quotes. Optional.

  For -ccflags and -ldflags, qverilog does not check the validity of the option(s) you specify. The options are directly passed on to the compiler and linker, and if they are not valid, an error message is generated by the compiler/linker.

- -ccwarn [on | off | verbose | strict]

  Echoes warnings generated during C/C++ source file compilation to stdout.

on — (default) compiles with the -Wreturn-type, -Wimplicit, -Wuninitialized, and -Wmissing-declarations gcc options.

off — compiles without any warning options.

verbose —compiles with the -Wall gcc option.

strict — compiles with the -Werror gcc option.

- -do <dofile>

  (optional) Simulates the design using the specified dofile.

- -gui

  (optional) Simulates the design using the ModelSim GUI.

- -i

  (optional) Specifies that the simulator be run in interactive mode.

- -l <logfile>

  Creates a logfile/transcript compatible with vlog -l logfile. Optional. If omitted, a default transcript called *qverilog.log* is created that collects the output from vlog, vopt, and vsim.

- <filename>

  Specifies the name of the Verilog or C/C++ source code file to compile. One filename is required. Multiple filenames separated by spaces may be entered. Wildcards may be used. In the case of C files, they are automatically processed as DPI code.

- -ldflags "opts"

  Specifies all linker options in quotes. Optional.

  For -ccflags and -ldflags, qverilog does not check the validity of the option(s) you specify. The options are directly passed on to the compiler and linker, and if they are not valid, an error message is generated by the compiler/linker.

- -R <vsim_options>

  Specifies valid vsim arguments to be applied to the simulation. All vlog and vopt arguments must come before -R is specified, as all arguments specified after -R are interpreted as vsim arguments.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  qverilog -stats options are passed to each sub-process (vcom, vlog, vopt, vsim) and add or subtract features and modes from the settings in the Stats *modelsim.ini* variable for each sub-process . The -stats argument is ignored if specified after -R. Statistics are collated from all sub-commands of qverilog and printed once for each execution of qverilog.

[+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

Features

all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

cmd — (default) Echo the command line.

msg — (default) Display error and warning summary at the end of command execution.

none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

perf — Display time and memory performance statistics.

time — (default) Display Start, End, and Elapsed times.

Modes

Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover dump -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover dump -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

kb — Print statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

Note

qverilog -quiet disables all default or user-specified -stats features.

- -work <library_name>

  Specifies a logical name or pathname of a library that is to be mapped to the logical library work. Optional. By default, the compiled design units are added to the work library. The specified pathname overrides the pathname specified for work in the project file.

Examples

- Compile, optimize, and simulate the specified files. The C/C++ code contained in the *d.c* file is processed as DPI code, creating a shared object, and loading it into vsim at runtime. Creates a logfile named "logfile" and opens the Questa SIM GUI with the simulation loaded and ready to run.

```
qverilog -l logfile a.v b.v c.v d.c -R -gui
```

- Enable the time, msg, and perf features and disable the cmd feature for vlog and vsim; ignore -stats=none,cmd to vsim (specified after -R).

```
qverilog -stats=time,-cmd,msg,perf file.v -R \
-stats=none,cmd -do "run 1; quit -f"
```

- The first -stats option is ignored. The none option disables all *modelsim.ini* settings and then enables the perf option.

```
qverilog -stats=time,cmd,msg -stats=none,perf
```

radix

This command specifies the default radix to be used for the current simulation. Specifying the command with no argument returns the current radix setting.

The command can be used at any time.

The specified radix is used for all commands (force, examine, change, etc.) as well as for displayed values in the Objects, Locals, Dataflow, Schematic, List, and Wave windows, as well as the Source window in the source annotation view.

Alternate methods for changing the default radix:

- In the *modelsim.ini* file, edit the DefaultRadix variable.
- Choose **Simulate** > **Runtime Options** from the main menu, click the **Defaults** tab, make your selection in the **Default Radix** box.

Syntax

radix [-binary | -octal | -decimal | -hexadecimal | -unsigned | -ascii | -time]
[-enumnumeric | -enumsymbolic | -showbase | -symbolic]

Arguments

You can abbreviate the following arguments to any length. For example, -dec is equivalent to -decimal.

- -ascii

  (optional) Display a Verilog object as a string equivalent using 8-bit character encoding.
- -binary

  (optional) Displays values in binary format.
- -enumnumeric

  (optional) Causes Verilog and SystemC enums to be displayed as numbers (formatted by the current radix). This overrides the default behavior of always showing enums symbolically.
- -enumsymbolic

  (optional) Restores the default behavior of displaying Verilog and SystemC enums as symbols by reversing the action of the -enumnumeric option.
- -decimal

  (optional) Displays values in decimal format. You can specify -signed as an alias for this argument.
- -hexadecimal

  (optional) Displays values in hexadecimal format.
- -octal

  (optional) Displays values in octal format.

- -time

  (optional) Displays values of time for register-based types in Verilog.

- -showbase

  (optional) Display the number of bits of the vector and the radix used, where:

  binary = b
  decimal = d
  hexidecimal = h
  ASCII = a
  time = t

  For example, instead of simply displaying a vector value of “31”, a value of “16’h31” may be displayed to show that the vector is 16 bits wide, with a hexidecimal radix.

- -symbolic

  (optional) Displays values in a form closest to their natural form.

- -unsigned

  (optional) Displays values in unsigned decimal format.

Description

Numeric radix other than symbolic are defined to translate bits to a 4 state representation as part of the numeric conversion. Groups of bits are then converted to a number in the correct radix, or to 'x' or 'z' if the value is not numeric (that is, contains only '0's and '1's). There is no 'U' in the 4 state representation, nor 'W' or '-'. All of the odd values are converted to 'x'.

Alternatives to changing the default radix for the simulation session include:

- use examine <name> -<radix_type> to transcript the current value of <name> using the specified radix
- use radix signal to set a signal-specific radix

Related Topics

- User-Defined Radices
- radix define
- radix delete
- radix names
- radix list
- radix signal

radix define

This command is used to create or modify a user-defined radix. A user definable radix is used to map bit patterns to a set of enumeration labels or setup a fixed or floating point radix. User-defined radices are available for use in the most windows and with the examine command.

Syntax

User Custom Radix

radix define **<name> <definition_body>** [-color <value>]

Fixed or Floating Point Number Radix

radix define **<name>** [[**-fixed** [-signed] | **-float**] **-fraction <n>**] [-base <base>] [-precision <p>]

Arguments

- <name>

  (required) User-specified name for the radix.

- <definition_body>

  (required for custom radix) A list of number pattern, label pairs. The definition body has the form:

```
{
   <numeric-value> <enum-label> [-color <color>],
   <numeric-value> <enum-label>
   -default <radix_type>
   -defaultcolor <color>
}
```

A <numeric-value> is any legitimate HDL integer numeric literal. To be more specific:

```
<integer>
<base>#<value>#        --- <base> is 2, 8, 10, or 16
<base>"value"          --- <base> is B, O, or X
<size>'<base><value>   --- <size> is an integer,
                           <base> is b, d, o, or h.
```

The question mark (?) wildcard character may be specified for bits or characters of the value. For example:

radix define bus-state {

6'b01??00 "Write" -color orange,

6'b10??00 "Read" -color green

}

In this example, the first pattern will match "010000", "010100", "011000", and "011100". In case of overlaps, the first matching pattern is used, going from top to bottom.

The comma (,) in the definition body is optional. The <enum-label> is any arbitrary string. It should surrounded by quotation marks ("), especially if it contains spaces.

The -default entry is optional. If present, it defines the radix to use if a match is not found for a given value. The -default entry can appear anywhere in the list, it does not have to be at the end. The optional -color argument in the definition body will set the color for the specific value when the value is displayed in the Wave window.

Refer to the Verilog and VHDL Language Reference Manuals for exact definitions of these numeric literals.

- -base <base>

  (optional for fixed and floating point radices) Specifies the base for a fixed or floating point radix.

  <base> — Any valid radix type: binary, ascii, unsigned, decimal, octal, hex, symbolic, time, and default.

- -color <value>

  (optional for custom radices) Designates a color for the waveform and text in the Wave window.

  <value> — The color value may be a color name or its hex value (see example below).

- -fixed

  (required for fixed point radix) Specifies a fixed number radix.

- -float

  (required for floating point radix) Specifies a floating point number radix.

- -fraction <n>

  (required for fixed and floating point radices) Specifies the location of the decimal point in a vector.

  <n> — Any integer between 3 and the full bit value of the vector. For example, if you specify -fraction 3 for the eight bit vector “10001001”, the decimal is placed two bits away from the least significant bit on the right so the vector becomes “10001.001”.

- -precision <p>

  (optional for fixed and floating point radices) Specifies now many numbers after the decimal point or significant digits of a floating point or fixed number in symbolic format.

  <p> — A number and optional format specification taking the form “<width>[efg],” for example 3g, 4f, or 6.

- -signed

  (optional for fixed point radix) Treats fixed numbers as signed where the most significant bit is the sign bit. The default is an unsigned number.

Examples

- The radix define command used to create a radix called “States,” which will display state values in the List, Watch, and Wave windows instead of numeric values.

```
radix define States {
    11'b00000000001 "IDLE",
    11'b00000000010 "CTRL",
    11'b00000000100 "WT_WD_1",
     11'b00000001000 "WT_WD_2",
     11'b00000010000 "WT_BLK_1",
     11'b00000100000 "WT_BLK_2",
    11'b00001000000 "WT_BLK_3",
    11'b00010000000 "WT_BLK_4",
    11'b00100000000 "WT_BLK_5",
    11'b01000000000 "RD_WD_1",
    11'b10000000000 "RD_WD_2",
    -default hex
}
```

- The following example illustrates how to specify the radix color:

```
radix define States {
    11'b00000000001 "IDLE" -color yellow,
    11'b00000000010 "CTRL" -color #ffee00,
    11'b00000000100 "WT_WD_1" -color orange,
    11'b00000001000 "WT_WD_2" -color orange,
    11'b00000010000 "WT_BLK_1",
    11'b00000100000 "WT_BLK_2",
    11'b00001000000 "WT_BLK_3",
    11'b00010000000 "WT_BLK_4",
    11'b00100000000 "WT_BLK_5",
    11'b01000000000 "RD_WD_1" -color green,
    11'b10000000000 "RD_WD_2" -color green,
    -default hex
    -defaultcolor white
}
```

If a pattern/label pair does not specify a color, the normal wave window colors will be used. If the value of the waveform does not match any pattern, then the -default radix and -defaultcolor will be used.

- Create a fixed point radix named fx5 and apply that radix to the signal checksf.

  Entering

  VSIM> radix define fx5 -fixed -fraction 3 -base decimal -signed

  returns

  ```
  #fx5
  ```

  Entering

  VSIM> radix signal checksf

  returns

  ```
  #fx5
  ```

  Entering

  VSIM> examine -name cecksf

  returns

  ```
  #/test_fixed/basictest/checksf -15.8750
  ```

Related Topics

- User-Defined Radices
- precision
- radix
- radix delete
- radix names
- radix list
- radix signal

radix delete

This command will remove the radix definition from the named radix.

Syntax

radix delete **<name>**

Arguments

- <name>

  (required) Removes the radix definition from the named radix.

Related Topics

- User-Defined Radices
- radix
- radix define
- radix list
- radix names
- radix signal

radix list

This command will return the complete definition of a radix, if a name is given. If no name is given, it will list all the defined radices.

Syntax

radix list [<name>]

Arguments

- <name>

  (optional) Returns the complete definition of the named radix.

Related Topics

- User-Defined Radices
- radix
- radix define
- radix delete
- radix names
- radix signal

radix names

This command returns a list of currently defined radix names.

Syntax

radix names

Arguments

None

Related Topics

- User-Defined Radices
- radix
- radix define
- radix delete
- radix list
- radix signal

radix signal

This command sets or inspects radix values for the specified signal in the Objects, Locals, Schematic, and Wave windows.

Note

The intent is for this command to be used for a small number of signals. If the majority of signals in a design are to use a particular radix value, then set that value as the default radix with the radix command, and use the radix signal command for the rest.

When no argument is used, the radix signal command returns a list of all signals with a radix.

Syntax

radix signal [<signal_name> [<radix_value>]] [-fpoint <decimal>] [-showbase]

Arguments

- <signal_name>

  (optional) Name of the signal for which the radix will be set (if <radix_value> is specified) or inspected.

- <radix_value>

  (optional) Value of the radix to be set for the specified signal. Use empty quotation marks ("") to unset the radix for the specified signal.

- -fpoint <decimal>

  (optional) Designates a fixed point radix with “decimal” specifying the number of decimal places of the radix.

- -showbase

  (optional) Display the number of bits of the vector and the radix used, where:

  binary = b
  decimal = d
  hexidecimal = h
  ASCII = a
  time = t

  For example, instead of simply displaying a vector value of “31”, a value of “16'h31” may be displayed to show that the vector is 16 bits wide, with a hexidecimal radix.

Related Topics

- User-Defined Radices
- radix
- radix define
- radix list
- radix delete

readers

This command displays the names of all readers of the specified object.

The reader list is expressed relative to the top-most design signal/net connected to the specified object.

The output from the readers command, which is displayed in the Transcript window as a hypertext link, allows you to right-click to open a drop-down menu and to quickly add signals to various windows. It includes a "View Declaration" item to open the source definition of the signal.

Figure 2-4. readers Command Results in Transcript

Syntax

readers <object_name> [-source]

Arguments

- <object_name>

  (required) Specifies the name of the signal or net whose readers are to be shown. All signal or net types are valid. Multiple names and wildcards are accepted.

- -source

  (optional) Returns the source file name and line number for each driver of the specified signal or net. If the source location cannot be determined, the value n/a is returned for that driver.

Related Topics

- drivers

report

This command displays information relevant to the current simulation.

Note

Viewing preference variables: preference variables have more to do with the way things look (but not entirely) rather than controlling the simulator. You can view preference variables from the Preferences dialog box. Select **Tools** > **Edit Preferences** (Main window).

Syntax

report files

report where [ini] [pwd] [transcript] [wlf] [project]

report simulator control

report simulator state

Arguments

- files

  Returns a list of all source files used in the loaded design. This information is also available in the Specified Path column of the Files window.

- where [ini] [pwd] [transcript] [wlf] [project]

  Returns a list of configuration files where the arguments limit the list to those files specified. If specified without arguments, returns a list of all configuration files in the current simulation.

  ini — (optional) Returns the location of the *modesim.ini* file.

  pwd — (optional) Returns the current working directory.

  transcript — (optional) Returns the location for saving the transcript file.

  wlf — (optional) Returns the current location for saving the *.wlf* file.

  project — (optional) Returns the current location of the project file.

- simulator control

  Displays the current values for all simulator control variables.

- simulator state

  Displays the simulator state variables relevant to the current simulation.

Examples

- Display configuration file information

  report where

  Returns:

```
# INI {modelsim.ini}
# PWD ./Testcases/
# Transcript transcript
# WLF vsim.wlf
# Project {}
```

- Display all simulator control variables.

  report simulator control

  Returns:

```
# UserTimeUnit = ns
# RunLength =
# IterationLimit = 5000
# BreakOnAssertion = 3
# DefaultForceKind = default
# IgnoreNote = 0
# IgnoreWarning = 0
# IgnoreError = 0
# IgnoreFailure = 0
# IgnoreSVAInfo= 0
# IgnoreSVAWarning = 0
# IgnoreSVAError = 0
# IgnoreSVAFatal = 0
# CheckpointCompressMode = 1
# NumericStdNoWarnings = 0
# StdArithNoWarnings = 0
# PathSeparator = /
# DefaultRadix = symbolic
# DelayFileOpen = 1
# WLFFilename = vsim.wlf
# WLFTimeLimit = 0
# WLFSizeLimit = 0
```

  You can set these simulator control variables to a new value by using the Tcl set Command Syntax.

- Display all simulator state variables. Only the variables that relate to the design being simulated are displayed:

  report simulator state

  Returns:

```
# now = 0.0
# delta = 0
# library = work
# entity = type_clocks
# architecture = full
# resolution = 1ns
```

Related Topics

- modelsim.ini Variables
- “Setting GUI Preferences”

restart

This command reloads the design elements and resets the simulation time to zero. Only design elements that have changed are reloaded. (Note that SDF files are always reread during a restart.)

- If no design is loaded, the restart command produces a message to that effect and takes no further action.
- If a simulation is loaded, the restart command restarts the simulation.
- If multiple datasets are open, including a simulation, the environment is changed to the simulation context and the simulation is restarted.

Shared libraries are handled as follows during a restart:

- Shared libraries that implement VHDL foreign architectures only are reloaded at each restart when the architecture is elaborated (unless vsim -keeploaded is used).
- Shared libraries loaded from the command line (-foreign and -pli options) and from the Veriuser entry in the *modelsim.ini* file are reloaded (unless vsim -keeploaded is used).
- Shared libraries that implement VHDL foreign subprograms remain loaded (they are not reloaded) even if they also contain code for a foreign architecture.

You can configure defaults for the restart command by setting the DefaultRestartOptions variable in the *modelsim.ini* file. Refer to "Restart Command Defaults".

To handle restarts with Verilog PLI applications, you need to define a Verilog user-defined task or function, and register a misctf class of callback.

To handle restarts with Verilog VPI applications, you need to register reset callbacks. To handle restarts with VHDL FLI applications, you need to register restart callbacks.

Refer to "Verilog Interfaces to C" for more information on the Verilog HDL interfaces.

Refer to the *Questa SIM FLI Reference* for more information on the FLI.

Syntax

restart [-force] [-nobreakpoint] [-nolist] [-nolog] [-nowave] [-noassertions] [-nofcovers]

Arguments

- -force

  (optional) Specifies that the simulation will be restarted without requiring confirmation in a popup window.

- -noassertions

  (optional) Specifies that current assert directive configurations will not be maintained after the simulation is restarted where the default is for assert directive settings to be maintained after the simulation is restarted.

- -nobreakpoint

  (optional) Specifies that all breakpoints will be removed when the simulation is restarted where the default is for all breakpoints to be reinstalled after the simulation is restarted.

- -nofcovers

  (optional) Specifies that current cover directive configurations will not be maintained after the simulation is restarted where the default is for cover directive settings to be maintained after the simulation is restarted.

- -nolist

  (optional) Specifies that the current List window environment will not be maintained after the simulation is restarted where the default is for all currently listed HDL objects and their formats to be maintained.

- -nolog

  (optional) Specifies that the current logging environment will not be maintained after the simulation is restarted where the default is for all currently logged objects to continue to be logged.

- -nowave

  (optional) Specifies that the current Wave window environment will not be maintained after the simulation is restarted where the default is for all objects displayed in the Wave window to remain in the window with the same format.

Related Topics

- checkpoint
- restore
- vsim
- Checkpointing and Restoring Simulations

restore

This command restores the state of a simulation that was saved with a checkpoint command during the current invocation of vsim (called a "warm restore").

The items restored are: simulation kernel state, *vsim.wlf* file, HDL objects listed in the List and Wave windows, file pointer positions for files opened under VHDL and under Verilog $fopen, and the saved state of foreign architectures.

If you want to restore while running vsim, use this command. If you want to start up vsim and restore a previously-saved checkpoint, use vsim -restore (called a "cold restore").

Checkpoint/restore allows a cold restore, followed by simulation activity, followed by a warm restore back to the original cold-restore checkpoint file. Warm restores to checkpoint files that were not created in the current run are not allowed except for this special case of an original cold restore file.

Restrictions for this command include:

- Checkpoint files are platform dependent—you cannot checkpoint on one platform and restore on another.
- The restore command is not supported once a SystemC design has been loaded.

Syntax

restore <pathname>

Arguments

- <pathname>

  (required) Specifies the name of the checkpoint file, or a directory pathname where the *vsim.cpt* checkpoint file is located.

Related Topics

- checkpoint
- vsim
- Checkpointing and Restoring Simulations

resume

This command is used to resume execution of a macro (DO) file after a pause command or a breakpoint.

This command may be input manually or placed in an onbreak command string. (Placing a resume command in a bp command string does not have this effect.) The resume command can also be used in an onerror command string to allow an error message to be printed without halting the execution of the macro file.

Syntax

resume

Arguments

None

Related Topics

- abort
- do
- onbreak
- onerror
- pause
- Useful Commands for Handling Breakpoints and Errors

right

This command searches right (next) for signal transitions or values in the specified Wave window.

It executes the search on signals currently selected in the window, starting at the time of the active cursor. The active cursor moves to the found location.

Use this command to move to consecutive transitions or to find the time at which a waveform takes on a particular value, or an expression of multiple signals evaluates to true. See the left command for related functionality.

The procedure for using right entails three steps:

1. Click on the desired waveform.
2. Click on the desired starting location.
3. Issue the right command. (The seetime command can initially position the cursor from the command line, if desired.)

   Returns: <number_found> <new_time> <new_delta>

Note

"Wave Window Mouse and Keyboard Shortcuts" are also available for next and previous edge searches. Tab searches right (next) and shift-tab searches left (previous).

Syntax

right [-expr {<expression>}] [-falling] [-noglitch] [-rising] [-value <sig_value>] [-window <wname>] [<n>]

Arguments

- -expr {<expression>}

  (optional) Searches the waveform display for an expression. When the search evaluates to a boolean true, the active cursor moves to the found location. The expression may involve more than one signal, but is limited to signals that have been logged in the referenced Wave window. A signal may be specified either by its full path or by the shortcut label displayed in the Wave window.

  {<expression>} — Any valid expression. See GUI_expression_format for the format of the expression. The expression must be placed within curly braces.

- -falling

  (optional) Searches for a falling edge on the specified signal if that signal is a scalar signal. If it is not a scalar signal, the option will be ignored.

- -noglitch

  (optional) Looks at signal values only on the last delta of a time step. For use with the -value option only.

- -rising

  (optional) Searches for a rising edge on the specified signal if that signal is a scalar signal. If it is not a scalar signal, the option will be ignored.

- -value <sig_value>

  (optional) Specifies a value of the signal to match. Only one signal may be selected, but that signal may be an array.

  <sig_value> — Must be specified in the same radix that the selected waveform is displayed. Case is ignored, but otherwise the value must be an exact string match -- don't-care bits are not yet implemented.

- -window <wname>

  (optional) Specifies an instance of the Wave window that is not the default. Otherwise, the default Wave window is used. Use the view command to change the default window.

  <wname> — The name of a Wave window other than the current default window.

- <n>

  (optional) Specifies to find the nth match. If less than n are found, the number found is returned with a warning message, and the cursor is positioned at the last match. The default is 1.

Examples

- Find the second time to the right at which the selected vector transitions to FF23, ignoring glitches.

  right -noglitch -value FF23 2

- Go to the next transition on the selected signal.

  right

The following examples illustrate search expressions that use a variety of signal attributes, paths, array constants, and time variables. Such expressions follow the GUI_expression_format.

- Search right for an expression that evaluates to a boolean 1 when signal *clk* just changed from low to high and signal *mystate* is the enumeration reading and signal */top/u3/addr* is equal to the specified 32-bit hex constant; otherwise is 0.

  right -expr {clk'rising && (mystate == reading) && (/top/u3/addr == 32'habcd1234)}

- Search right for an expression that evaluates to a boolean 1 when the upper 8 bits of the 32-bit signal */top/u3/addr* equals hex ac.

  right -expr {(/top/u3/addr and 32'hff000000) == 32'hac000000}

- Search right for an expression that evaluates to a boolean 1 when logfile time is between 23 and 54 microseconds, and clock just changed from low to high and signal *mode* is enumeration writing.

  right -expr {((NOW > 23 us) && (NOW < 54 us)) && clk'rising && (mode == writing)}

Related Topics

- GUI_expression_format
- left
- seetime
- view

run

This command advances the simulation by the specified number of timesteps.

Syntax

run {[<timesteps>[<time_units>]] | -all | -continue | -final | -finish | -init | -next } |
{-step [-current] [<n>] [-out] [-over [<n>]] [-this "this==<class_handle>"]}

Arguments

- No arguments

  Runs the simulation for the default time (100 ns).

  You can change the default <timesteps> and <time_units> in the GUI with the Run Length toolbar box in the Simulate toolbar or from the *modelsim.ini* file: RunLength and UserTimeUnit variables.

- <timesteps>[<time_units>]

  (optional) Specifies the number of timesteps for the simulation to run. The number may be fractional, or may be specified as absolute by preceding the value with the character @.

  <time_units> — Any valid time unit: fs, ps, ns, us, ms, or sec where the default is to use the current time unit.

- -all

  (optional) Causes the simulator to run the current simulation forever, or until it hits a breakpoint or specified break event.

- -continue

  (optional) Continues the last simulation run after a run -step, run -step -over command or a breakpoint. A run -continue command may be input manually or used as the last command in a bp command string.

- -final

  (optional) Instructs the simulator to run all final blocks then exit the simulation.

- -finish

  (optional) In C Debug only, continues the simulation run and returns control to the calling function.

- -init

  (optional) Initializes non-trivial static SystemVerilog variables before beginning the simulation, for example, expressions involving other variables and function calls,. This could be useful for when you want to initialize values before executing any force, examine, or bp commands.

  You cannot use run -init after any other run commands or if you have specified vsim -runinit on the command line because all variables would have been initialized by that point.

- -next

  (optional) Causes the simulator to run to the next event time.

- -step

  (optional) Steps the simulator to the next HDL or C statement.

  Current values of local HDL variables may be observed at this time using the Locals window. You can specify the following arguments when you use -step:

  -current

  (optional) Instructs the simulation to step into an instance, process, or thread and stay in the current thread. Prevents stepping into a different thread.

  <n>

  (optional) Moves the simulator <n> steps ahead. Moves the debugger <n> lines ahead when you are using C Debug. Specified as a positive integer value.

  -out

  (optional) Instructs the simulation to step out of the current function or procedure and return to the caller.

  -over

  Directs Questa SIM to run VHDL procedures and functions, Verilog tasks and functions, and C functions but to treat them as simple statements instead of entering and tracing them line by line.

  You can use the -over argument to skip over a VHDL procedures or functions, Verilog task or functions, or a C function.

  When a wait statement or end of process is encountered, time advances to the next scheduled activity. Questa SIM then updates the Process and Source windows to reflect the next activity.

  -this "this==<class_handle>"

  (optional) Instructs the simulation to step into a method of a SystemVerilog class when “this” refers to the specified class handle. To obtain the handle of the class, use the examine -handle command.

  <class_handle> — Specifies a SystemVerilog class. Note that you must use quotation marks (" ") with this argument.

Description

You can control any return values after the run operation completes with the following preference variables:

- noRunTimeMsg — Set this variable to 0 to display simulation time and delta information or set it to 1 to disable the display of this information.
- noRunStatusMsg — Set this variable to 0 to display run status informatin or set it to 1 to disable the display of this information.

The following is an example that shows a series of run commands and how the output changes with the preference variable settings:

```
VSIM 1> run 105
VSIM 2> set PrefMain(noRunTimeMsg) 0
# 0

VSIM 3> run 112
# Time: @217 ns 0

VSIM 4> set PrefMain(noRunStatusMsg) 0
# 0

VSIM 5> run 100
# Time: @317 ns 0
# Status: ready end

VSIM 6> set PrefMain(noRunTimeMsg) 1
# 1

VSIM 7> run 50
# Status: ready end

VSIM 8> set PrefMain(noRunStatusMsg) 1
# 1

VSIM 9> run 55

VSIM 10>
```

Examples

- Advance the simulator 1000 timesteps.

  ```
  run 1000
  ```

- Advance the simulator the appropriate number of timesteps corresponding to 10.4 milliseconds.

  ```
  run 10.4 ms
  ```

- Advance the simulator to timestep 8000.

  ```
  run @8000
  ```

- Advance the simulator into the instance /top/p.

  ```
  run -step -current /top/p
  ```

Related Topics

- Simulate Toolbar
- C Debug
- step

runStatus

This command returns the current state of your simulation to stdout after issuing a run or step command.

Syntax

runStatus [-full]

Arguments

- -full

  (optional) Appends additional information to the output of the runStatus command.

Returns

Table 2-6 (runStatus Command States) and Table 2-7 (runStatus -full Command Information) show outputs of the runStatus command.

Table 2-6. runStatus Command States

State	Description
ready	The design is loaded and is ready to run.
break	The simulation stopped before completing the requested run.
error	The simulation stopped due to an error condition.
loading	The simulation is currently elaborating.
nodesign	There is no design loaded.
checkpoint	A checkpoint is being created, do not interrupt this process.
cready	The design is loaded and is ready to run in C debug mode.
initializing	The user interface initialization is in progress.

Table 2-7. runStatus -full Command Information

-full Information	Description
bkpt	stopped at breakpoint
bkpt_builtin	stopped at breakpoint on builtin process
end	reached end of requested run
fatal_error	encountered fatal error (such as, divide by 0)
iteration_limit	iteration limit reached, possible feedback loop
silent_halt	mti_BreakSilent() called,
step	run -step completed
step_builtin	run -step completed on builtin process

Table 2-7. runStatus -full Command Information (cont.)

-full Information	Description
step_wait_suspend	run -step completed, time advanced.
user_break	run interrupted do to break-key or ^C (SIGINT)
user_halt	mti_Break() called.
user_stop	stop or finish requested from stop command, etc.
gate_oscillation	Verilog gate iteration limit reached.
simulation_stop	pli stop_simulation() called.

sccom

The sccom command actually provides two different functions: sccom uses an external C/C++ compiler to compile SystemC source code into the work library, while sccom -link takes compiled source code and links the design.

The compilation of multiple SystemC source files can be distributed across multiple machines, or across multiple cores on a single machine using the -j argument. Distributed sccom compilation compiles source files in parallel on multiple cores of single or multiple machines, as specified by the user. The result is faster compilation.

Syntax

Compile syntax

sccom [-32 | -64] [-93] [<CPP compiler options>] [<CPP linker options>]
[-cppinstall <[gcc|g++] version>] [-cpppath <filename>] [-dumpscvext <filename>]
[-dpilib <libname>] [-error <msg_number> [,<msg_number>,...]]
[[-F | -file | -f] <filename>] [-fatal <msg_number>[,<msg_number>,…]]
[-gnuversion] [-help] [-incr] [-j <value>]
[-lib <library_pathname>] [-libshared <library_pathname>] [-link] [-linkshared]
[-log <logfile>] [-machines <hosts.txt>] [-modelsimini <path/modelsim.ini>]
[-nodbgsym] [-nodebug] [-nologo] [-note <msg_number> [,<msg_number>,...]]
[-optionset <optionset_name>] [-predefmacrofile <filename>] [-sc22] [-scms] [-scv]
[-scversion] [-stats [=[+ | -]<arg>[,[+ | -]<arg>]]
[-suppress <msg_number> [,<msg_number>,...]] [-vv] [-verbose] [-version]
[-warning <msg_number> [,<msg_number>,...]] [-work <library_name>] [-x c | c++]
<filename>

Link syntax

sccom -link

[<CPP linker options>] [-dpilib <libname>] [[-F | -file | -f] <filename>] [-help]
[-lib <library_pathname>] [-log <logfile>] [-nologo] [-sc22] [-scv] [-vv] [-verbose]
[-version] [-work <library_name>]

Arguments

- -32 | -64

  Specifies whether sccom uses the 32- or 64-bit executable, where -32 is the default.

  These options apply only to the supported Linux operating systems.

  These options override the MTI_VCO_MODE environment variable, which applies only to executables used from the *<install_dir>/bin/* directory. Therefore, these options are ignored if you run sccom from an *<install_dir>/<platform>/* directory.

  You can only specify these options on the command line, therefore they are not recognized as part of a file used with the -f switch.

- -93

  (optional) Makes the design unit strictly case-sensitive, enforcing case-sensitivity across the SystemC-HDL mixed language boundary.

- <CPP compiler options>

  Any normal C++ compiler option can be used, with the exception of the -o and -c options. You must specify the -g argument to compile for debugging. By default, sccom compiles without debugging information. You can specify arguments for all sccom compiles by editing the CppOptions variable in the *modelsim.ini* file.

  - -DSC_

    (optional) Specifies SystemVerilog libraries for SystemC DPI (Direct Programming Interface). Only one library can be specified per each -dpilib argument. See section SystemC Procedural Interface to SystemVerilog.

  - -DSC_INCLUDE_MTI_AC

    (optional) Enable native debug support of Algorithmic-C datatypes.

  - -DSC_INCLUDE_DYNAMIC_PROCESSES

    (optional) Enable dynamic processes.

  - -DSC_INCLUDE_FX

    (optional) Enable fixed-point datatypes.

  - -DSC_USE_STD_STRING

    (optional) Replace sc_string with std::string.

  - -DSC_USE_STD_STRING_OLD

    (optional) Use deprecated sc_string.

  - -DMTI_BIND_SC_MEMBER_FUNCTION

    (optional) Enable registration of module member functions as DPI-SC imports.

  - -DUSE_MTI_CIN

    (optional) Enables support of C++ standard input *cin*.

- <CPP linker options>

  Any normal C++ compiler option can be used, with the exception of the -o option. You can specify arguments for all sccom compiles by editing the CppOptions variable in the *modelsim.ini* file.

- -cppinstall <[gcc|g++] version>

  (optional) Specifies the version of the desired GNU compiler supported and distributed by Mentor Graphics.

<[gcc|g++] version> — The version number of the GNU compiler to use. For example:

```
sccom -cppinstall 4.5.0
```

- -cpppath <filename>

  (optional) Specifies the location of a g++ executable other than the default g++ compiler installed with Questa SIM. Overrides the CppPath variable in the *modelsim.ini* file.

- -dumpscvext <filename>

  (optional) Generates SystemC verification (SCV) extensions for any given object type. For this argument, <filename> is a C++ (.cpp) file that contains global variable definition for each type and includes the header file containing definitions for these types.

- -dpilib <libname>

  (optional) Specifies SystemVerilog libraries for SystemC DPI (Direct Programming Interface). Only one library can be specified per each -dpilib argument. See section SystemC Procedural Interface to SystemVerilog.

- -error <msg_number> [,<msg_number>,...]

  (optional) Changes the severity level of the specified message(s) to "error." Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information.

- [-F | -file | -f] <filename>

  (optional) -f, -file and -F: each specifies an argument file with more command-line arguments, allowing complex argument strings to be reused without retyping. Nesting of -F, -f and -file commands is allowed. Allows gzipped input files.

  With -F only: relative file names and paths within the arguments file <filename> are prefixed with the path of the arguments file when lookup with relative path fails. Refer to the section "Argument Files" for more information.

- -fatal <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "fatal." Edit the fatal variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information.

- -gnuversion

  Returns the version and path of the GNU compiler used for compilation. You can specify this argument by itself on the sccom command line. Refer to the section “Verifying Compiler Information” in the User’s Manual.

- -help

  Displays options and arguments for this command. Optional.

- -incr

  (optional) Enables automatic incremental compilation so that only changed files are compiled. A changed file is re-compiled in the following cases:

- Its pre-processor output is different from the last time it was successfully compiled. This includes changes in included header files and to the source code itself.

Note

Pre-processor output is used because it prevents compilation on a file with the following types of changes: Access or modification time (touch) or Changes to comments—except changes to the source code that affect line numbers (such as adding a comment line) will cause all affected files to be recompiled. This occurs to keep debug information current so that Questa SIM can trace back to the correct areas of the source code.

- You invoke sccom with a different set of command-line options that have an impact on the gcc command line. Preserving all settings for the gcc command ensures that Questa SIM re-compiles source files when a different version of gcc is used or when a platform changes.

- -j <value>

  (required for distributed compile) Specifies the maximum number of parallel compilation processes requested. <value> should be a positive integer greater than or equal to zero. If <value> is 0, or -j is not specified, sccom runs in the default non-distributed mode.

- -lib <library_pathname>

  Specifies the default working library where the SystemC linker can find the object files for compiled SystemC modules. Use only with sccom -link. Optional.

- -libshared <library_pathname>

  (UNIX, Linux only) Specifies the library location for the intermediate SystemC library specified with sccom -linkshared. This argument cannot be used with sccom -link. Optional.

- -link

  (required before running simulation) Performs the final link of all previously compiled SystemC source code. You must specify any dependent *.a* or *.o* before the *.a* or *.o* on which it depends. Two types of dependencies are possible, and where you place the -link argument is different based on which type of dependency the files have.

  If your archive or object is dependent on the *.o* files created by sccom (that is, your code references symbols in the generated SystemC *.o* files), then you must specify the -link argument after the list of files, as follows:

  ```
  sccom a.o b.o libtemp.a -linkz
  ```

  Functionally, the order of the C++ linker command and argument looks like this:

  ```
  ld a.o b.o libtemp.a <internal list of SC .o files> libsystemc.a
  ```

  However, if the *.o* files created by sccom are dependent on the object or archive you provided, then you must place the -link argument before the object files or archive:

```
sccom -link a.o b.o libtemp.a
```

In this case, the "functional" command and argument order look like this:

```
ld <internal list of SC .o files> libsystemc.a a.o b.o libtemp.a
```

- -linkshared

  (optional) (UNIX, Linux only) Creates an intermediate SystemC shared library, which allows you to create multiple SystemC shared libraries (systemc.so). This argument cannot be used with sccom -link.

- -log <logfile>

  (optional) Specifies the logfile in which to collect output. Related *modelsim.ini* variable is SccomLogfile.

- -machines <hosts.txt>

  (optional) Specifies a file containing a list of host machines to distribute parallel compilation processes. Used in conjunction with -j option to distribute total number of requested parallel processes evenly on given host machines. In the absence of this option, parallel compilation processes, as specified by the -j option, are started on the same machine. The format of entries in the <hosts.txt> file is:

  <machine_name>[:<int>]
  ...

  where <machine_name> is the name of an accessible machine to the one on which the sccom command is run, and <int> is the maximum number of processes on that machine. One entry allowed per line.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable. Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -nodbgsym

  (optional) Disables the generation of symbols for the debugging database in the library, which allows source annotation.

- -nodebug

  (optional) Disables the creation of a debug database for modules defined in those source files. Do not use this argument with any files containing the SC_MODULE_EXPORT() macro. Using -nodebug can significantly reduce the compile time for a design, which is useful when running designs in regression mode. Refer to "Custom Debugging of SystemC Channels and Variables" for more information.

- -nologo

  (optional) Disables the startup banner.

- -note <msg_number> [,<msg_number>,...]

  (optional) Changes the severity level of the specified message(s) to "note." Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -optionset <optionset_name>

  (optional) Calls an optionset as defined in the *modelsim.ini file*. Refer to the section "Optionsets" in the Reference Manual for more information.

- -predefmacrofile <filename>

  (optional) Specifies path to the predefined macros file. Generally, if you are using an unsupported g++ compiler, sccom generates a file "predef_macro_gcc<COMPILER_VERSION>_<PLATFORM>.txt" file, which must then be copied over to the location *$MTI_HOME/include/systemc*. Use of -predefmacrofile avoids the necessity of copying over this generated predefined macros file. Whether you copy the macros over or use -predefmacrofile, you must run sccom again for the predefined macros to be picked up.

- -sc22

  (optional) Enables SystemC-2.2 (the IEEE 1666-2005 standard) for both compiling and linking. You may also enable SystemC-2.2 with the Sc22Mode *modelsim.ini* variable.

- -scms

  (optional) Includes the SystemC master slave library. If you specify this argument when compiling your C code with sccom, you must also specify it when linking the object files with sccom -link.

- -scv

  (optional) Includes the SystemC verification library. If you specify this argument when compiling your C code with sccom, you must also specify it when linking the object files with sccom -link. Related *modelsim.ini* variable is UseScv.

- -scversion

  (optional) Prints out the version of the SystemC verification library used.

- -stats [=[+ | -]<arg>[,[+ | -]<arg>]

  (optional) Controls display of compiler statistics sent to a logfile, stdout, or the transcript. Specifying -stats without any options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

Arguments:

all | none— Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all or none is applied first.

cmd — (default) Echo the command line.

kb — Print statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

msg — (default) Display error and warning summary at the end of command execution.

perf — Display time and memory performance statistics.

time — (default) Display Start, End, and Elapsed times.

verbose — Display verbose statistics information when available.

Note

You can disable display of all statistics during compile by specifying -quiet.

- -suppress <msg_number> [,<msg_number>,...]

  (optional) Prevents the specified message(s) from displaying. You cannot suppress Fatal or Internal messages. Edit the suppress variable in the *modelsim.ini* file to set a permanent default.

- -vv

  (optional) Prints all subprocess invocation information. An example is the call to gcc along with the command-line arguments.

- -verbose

  (optional) Prints the name of each sc_module encountered during compilation. Related *modelsim.ini* variable SccomVerbose.

- -version

  (optional) Displays the version of sccom used to compile the design.

- -warning <msg_number> [,<msg_number>,...]

  (optional) Changes the severity level of the specified message(s) to "warning." Edit the warning variable in the *modelsim.ini* file to set a permanent default.

- -work <library_name>

  For the compiler — Specifies a logical name or pathname of a library that is to be mapped to the logical library work. Optional; by default, the compiled object files (.so) are added to the work library. The specified pathname overrides the pathname specified for work in the project file.

For the linker — Specifies a logical name or pathname of a library where the final linked object file (.so) is to be stored. Optional; by default, the linked object files are added to the work library.

- -x c | c++

  (optional) Specifies the language (C or C++) of a source file being compiled when the filename extension does not match that source. For example, if myfile.cpp contained C source, you would enter sccom -x c myfile.cpp. If you use this argument, you cannot combine C and C++ files with the same sccom command.

- <filename>

  (required) Specifies the name of a file containing the SystemC/C++ source to be compiled. You can enter multiple filenames separated by spaces; you can also use wildcards to specify multiple filenames (such as **.cpp*).

Description

You can run this command from within Questa SIM, from the operating system command prompt, or during simulation.

To enable source debugging of SystemC code, you must compile for debugging by specifying the -g argument of the CPP compiler.

Compiled libraries have the following dependencies:

- Platform — If you move between platforms, you need to run vdel -allsystemc on the working library and then recompile your SystemC source.
- Version — If you install a new release, you need to re-compile your library with the current version of sccom. For example, you cannot use a library compiled with v6.5 in a simulation using v6.5a vsim. You would have to run sccom in v6.5a to re-compile your library (sccom -version displays the version number of the compiler).

During the linking of the design (with sccom -link), the order in which you specify archives (*.a*) and object files is very important. You must specify any dependent *.a* or *.o* before the *.a* or *.o* on which it depends.

The sccom command can recognize the file type as either C or C++ by the filename extension and will use the appropriate compiler, as follows:

- gcc compiler on source files with C source extensions: *.c*, *.i*
- g++ compiler on source files with C++ extensions: .CPP , *.cpp* , *.C* , *.c++* , *.cc* , *.cp* , *.cxx*, *.ii*

For best performance, it is recommended to run sccom in multi-file compilation mode, which requires that you can write to the current working directory. By default, sccom works in multi-file compilation mode, passing all source files to the GNU compilers and debug generator in a single step. If the working directory has read-only permissions, sccom automatically performs single-file compilation, which decreases performance because only one source file is compiled at a time.

Examples

- Compile *example.cpp* with debugging information.

```
sccom -g example.cpp
```

- Link *example.o*.

```
sccom -link
```

- Use two sccom commands - the first to compile *a.cpp* into library *LIB_A*, and the second to compile *b.cpp* into *LIB_B*. (*a.cpp* defines a module, *TOP_A*, and *b.cpp* defines a module, *TOP_B*.) Run sccom again to link and compile the compiled object files created in those two libraries into a third shared library, *LIB_C*. Run vsim using -sclib, which is required in order to point to the location of the shared library.

```
vlib /path/to/LIB_A
vmap LIB_A /path/to/LIB_A
vlib /path/to/LIB_B
vmap LIB_B /path/to/LIB_B

sccom -work LIB_A a.cpp  -> a.o created in /path/to/LIB_A
sccom -work LIB_B b.cpp  -> b.o created in /path/to/LIB_B
```

  At this point you have the option to create the SystemC library in *LIB_A* or *LIB_B* or in a totally new library *LIB_C*.

  Include all objects from *LIB_B* and *LIB_A* and create a *.so* (shared object) in *LIB_A*:

```
sccom -link -work LIB_A -lib LIB_B
```

  Include all objects created in *LIB_A* and *LIB_B* and *LIB_C* and create a *.so* in *LIB_C*.

```
sccom -link -work LIB_C -lib LIB_A -lib LIB_B
```

  If the shared object is not compiled in the same library as the top-level design unit, the *.so* library has to be specified using the -sclib switch with the vsim command. The -lib switch tells the simulator where the top-level module is compiled. The vsim command also has a -L switch that allows you to specify the library where lower level modules can be found. For example:

```
vsim -sclib LIB_C -lib LIB_A TOP_A
```

  loads a SystemC shared library from *LIB_C*. The top module *TOP_A*, is compiled in *LIB_A*.

```
vsim -lib LIB_A TOP_A
```

  loads the SystemC shared library from *LIB_A* and the top module *TOP_A* from *LIB_A*.

```
vsim -lib LIB_A -sclib LIB_C -L LIB_B TOP_A
```

loads the SystemC shared library from *LIB_C*. The top level module, *TOP_A*, was compiled in *LIB_A*. *TOP_B*, which is instantiated in some hierarchy, can be found in *LIB_B*.

The vsim command can accept multiple -L switches, but it takes only one -lib switch. The -lib switch is for top-level modules and -L is for lower modules.

The -sclib switch specifies where the SystemC shared library was created.

- Compile the SystemC code with an include directory and the compile time macro (SC_INCLUDE_FX) to compile the source with support for fixed point types. For more information, refer to "Fixed-Point Types".

```
sccom -I/home/systemc/include -DSC_INCLUDE_FX -g a.cpp b.cpp
```

- Compile with the g++ -02 optimization argument.

```
sccom -02 a.cpp
```

- Link in the library *libmylib.a* when creating the *.so* file. The -L, a gcc argument, specifies the search path for the libraries.

```
sccom -L home/libs/ -l mylib -link
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
sccom -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all *modelsim.ini* settings and then enables the perf option.

```
sccom -stats=time,cmd,msg -stats=none,perf
```

Related Topics

- "SystemC Simulation"
- scgenmod
- vdel -allsystemc

scgenmod

Once a Verilog or VHDL module is compiled into a library, you can use the scgenmod command to write its equivalent SystemC foreign module declaration to standard output.

Optional -map argument allows you to appropriately generate sc_bit, sc_bv, or resolved port types; sc_logic and sc_lv port types are generated by default.

Syntax

scgenmod [-help] [-lib <library_name>] [-map "<hdl_type>=<sc_type>"] [-modelsimini <ini_filepath>] [-createtemplate] [-preservevhdlcase] <module_name>

Arguments

- -createtemplate

  (optional) Creates a class template declaration of a foreign module with integer template arguments corresponding to the integer parameter/generic defined in the VHDL or Verilog module. Ports in VHDL and Verilog modules instantiated from SystemC can now have their range specified in terms of integer parameters/generics. Such port ranges will be specified in terms of the template arguments of the foreign module.

- -help

  (optional) Displays the command's options and arguments.

- -lib <library_name>

  (optional) Specifies the pathname of the working library. If not specified, the default library work is used.

- -map "<hdl_type>=<sc_type>"

  (optional) Specifies the user defined type mappings between either SystemVerilog or VHDL and SystemC types. <hdl_type> is the supported SystemVerilog or VHDL types. <sc_type> is the supported SystemC types. No spaces are allowed.

  - SystemVerilog supported types:
    scalar (Verilog scalar), vector (Verilog vector), bit, logic, reg, bitvector (signed/unsigned, packed/unpacked bit vector), logicvector (signed/unsigned, packed/unpacked logic vector), regvector (signed/unsigned, packed/unpacked logic vector), integer, integer unsigned, int, int unsigned, shortint, shortint unsigned, longint, longint unsigned, byte, byte unsigned, struct (including fields with multi-dimensional arrays)

Note

You can map Verilog vectors to the following native C++ integer types:
short, int, long long.

- VHDL supported types:
  bit, bit_vector, enum, record (including nested records and fields with multi-dimensional arrays), std_logic, std_logic_vector, vl_logic, vl_logic_vector
- SystemC supported types:
  bool, sc_bit, sc_logic, sc_bv, sc_lv, short, unsigned short, int, unsigned int, long, unsigned long, long long, unsigned long long, sc_int, sc_signed, sc_unsigned, sc_bigint, sc_biguint, char, unsigned char

Note

VHDL/SystemVerilog ports of type multi-dimensional array get converted to SystemC signal arrays in the generated foreign module declaration.

- -modelsimini <ini_filepath>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable. Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).
- -preservevhdlcase

  (optional) Preserves the case for VHDL identifiers. By default, the scgenmod command converts all VHDL identifiers to lower case.
- <module_name>

  (required) Specifies the name of the Verilog/VHDL module to be accessed.

Examples

- This example uses a Verilog module that is compiled into the work library. The module begins as Verilog source code:

```
module vcounter (clock, topcount, count);
   input clock;
   input topcount;
   output count;
   reg count;
   ...
endmodule
```

- After compiling using vlog, you invoke scgenmod on the compiled module with the following command:

```
scgenmod vcounter
```

  The SystemC foreign module declaration for the above Verilog module is:

```
class vcounter : public sc_foreign_module
{
   public:
   sc_in<sc_logic> clock;
   sc_in<sc_logic> topcount;
   sc_out<sc_logic> count;
vcounter(sc_module_name nm),
   : sc_foreign_module(nm, hdl_name),
      clock("clock"),
      topcount("topcount"),
      count("count")
      {elaborate_foreign_module(hdl_hame);}
   ~vcounter()
   {}
};
```

Related Topics

- "SystemC Simulation"
- sccom

sdfcom

This command compiles the specified SDF file. Annotation of compiled SDF files can dramatically improve performance (compared to annotating the ASCII version of the same) in cases where the same SDF file is used for multiple simulation runs.

The compiled SDF file is annotated so that it is compatible with the vsim -v2k_int_delays command (that is, the annotator operates as if the vsim -v2k_int_delays command has been given).

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

sdfcom [-delayscale <value>] [-maxdelays] [-mindelays] [-nocompress] [-typdelays] **<source_file>** <output_file>

Arguments

- -delayscale <value>

  (optional) Scales delays by the specified value.

  <value> —

  If you use this argument during SDF compilation, do *not* use the scaling option when reading in the SDF file with vsim. If you do so, the delays will be scaled twice.

- -maxdelays

  (optional) Selects maximum delays from SDF delay values of the form (min:typ:max).

- -mindelays

  (optional) Selects minimum delays from SDF delay values of the form (min:typ:max).

- -nocompress

  (optional) Produces a compiled file that is not compressed with gzip. By default the compiled file is compressed with gzip (even though the resulting file does not have the usual *.gz* extension).

- -typdelays

  (optional) Selects typical delays from SDF delay values of the form (min:typ:max). Default.

- <source_file>

  (required) Specifies the SDF file to compile. Must be specified immediately in front of the <output_file> argument to the sdfcom command.

- <output_file>

  (required) Specifies the name for the compiled SDF file. Must be specified as the final argument to the sdfcom command.

Related Topics

- vsim -sdftyp
- Compiling SDF Files

search

This command searches the specified window for one or more objects matching the specified pattern(s). The search can be continued using the next command.

The search starts at the object currently selected, if any; otherwise it starts at the window top. The default action is to search downward until the first match, then move the selection to the object found, and return the index of the object found.

Returns the index of a single match, or a list of matching indexes. Returns nothing if no matches are found.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

search **<window_name>** [-window <wname>] [-all] [-field <n>] [-toggle] [-forward | -backward] [-wrap | -nowrap] [-exact] [-regexp] [-nocase] [-count <n>] **<pattern>**

Arguments for all windows

- <window_name>

  (required) Specifies the window in which to search. Can be one of:

  list, locals, objects, process, source, structure, or wave

  or a unique abbreviation. Must be specified as the first argument to the search command.

- -window <wname>

  (optional) Specifies a particular window when multiple instances of the window exist. If no window is specified the default window is used; the default window is determined by the most recent invocation of the view command and has " - Default" appended to the name.

  <wname> — Specifies the instance of the window to use (for example, wave2).

- -forward

  (optional) Search in the forward direction. (default)

- -backward

  (optional) Search in the reverse direction.

- <pattern>

  (required) String or glob-style wildcard pattern. Must be specified as the final argument to the search command.

Arguments, for all EXCEPT the Source window

- -all

  (optional) Finds all matches and returns a list of the indexes of all objects that match.

- -field <n>

  (optional) Selects different fields to test, depending on the window type:

Table 2-8. Field Arguments for Window Searches

Window	n=1	n=2	n=3	default
structure	instance	entity/module	architecture	instance
signals	name	-	cur. value	name
process	status	process label	fullpath	fullpath
variables	name	-	cur. value	name
wave	name	-	cur. value	name
list	label	fullname	-	label

  Default behavior for the List window is to attempt to match the label and if that fails, try to match the full signal name.

- -toggle

  (optional) Adds objects found to the selection. Does not do an initial clear selection. Otherwise deselects all and selects only one object.

- -wrap

  (optional) Specifies that the search continue from the top of the window after reaching the bottom. (default)

- -nowrap

  (optional) Specifies that the search stop at the bottom of the window and not continue searching from the top.

Arguments, Source window only

- -exact

  (optional) Search for an exact match.

- -regexp

  (optional) Use the pattern as a Tcl regular expression.

- -nocase

  (optional) Ignore case where the default is to use case.

- -count <n>

  (optional) Search for the nth match where the default is to search for the first match.

Description

With the -all option, the entire window is searched, the last object matching the pattern is selected, and a Tcl list of all corresponding indexes is returned.

With the -toggle option, objects found are selected in addition to the current selection.

For the List window, the search is done on the names of the objects listed, that is, across the header. To search for values of objects in the List window, use the down and up commands. Likewise, in the Wave window, the search is done on object names and values in the values column. To search for object values in the waveform pane of the Wave window, use the right and the left commands. You can also select **Edit > Search** in both windows.

Related Topics

- find
- next
- view

searchlog

This command searches one or more of the currently open logfiles for a specified condition.

It can be used to search for rising or falling edges, for signals equal to a specified value, or for when a generalized expression becomes true.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

searchlog [-command <cmd>] [-count <n>] [-deltas] [-endtime <time> [<unit>]] [-env <path>] [-event <time>] [-expr {<expr>}] [-reverse] [-rising | -falling | -anyedge] [-startDelta <num>] [-value <string>] **<startTime> [<unit>] <pattern>**

Description

If at least one match is found, it returns the time (and, optionally, delta) at which the last match occurred and the number of matches found, in a Tcl list:

```
{{<time>} <matchCount>}
```

where <time> is in the format <number> <unit>. If the -deltas option is specified, the delta of the last match is also returned:

```
{{<time>} <delta> <matchCount>}
```

If no matches are found, a TCL_ERROR is returned. If one or more matches are found, but less than the number requested, it is not considered an error condition, and the time of the farthest match is returned, with the count of the matches found.

Arguments

- -command <cmd>

  (optional) Specifies a Tcl command that will be called for each event on the specified signal.

  <cmd> — A Tcl command that receives four arguments and returns one of three values: "continue", "stop", or "" (empty).

  The command will be passed four arguments: the reason for the call, the time of the event, the delta for the event, and the value. The reason value will be one of WLF_STARTLOG, WLF_ENDLOG, WLF_EVENT, or WLF_WAKEUP. The function is expected to return "continue", "stop", or "" (empty). If "continue" or "" (empty) is returned, the search continues, making additional callbacks as necessary. If "stop" is returned, the search stops and control is returned to the caller of the searchlog command.

  Only searching for a single signal is supported.

- -count <n>

  (optional) Specifies to search for the nth occurrence of the match condition.

  <n> — Any positive integer.

- -deltas

  (optional) Indicates to test for a match on simulation delta cycles. Otherwise, matches are only tested for at the end of each simulation time step.

- -endtime <time> [<unit>]

  (optional) Specifies the simulation time at which to end the search. By default no end time is specified. Must be specified after -reverse when searching backwards.

  <time> — Specified as an integer or decimal number. Current simulation units are the default unless specifying <unit>.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid VHDL time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

  For -reverse searches the specified end time must be earlier than the specified <startTime>. For -reverse searches the default <endTime> is 0.

- -env <path>

  (optional) Specifies a path to a design region. Wildcards not allowed.

- -event <time>

  (optional) Indicates to test for a match on a simulation event. Otherwise, matches are only tested for at the end of each simulation time step.

- -expr {<expr>}

  (optional) Specifies a search for a general expression of signal values and simulation time. searchlog will search until the expression evaluates to true.

  {<expr>} — An expression that evaluates to a boolean true. See GUI_expression_format for the format of the expression.

- -reverse

  (optional) Specifies to search backwards in time from <startTime>. You can limit the time span for the reverse search by including the -endtime <time> argument.

- -rising

  (optional) Specifies a search for rising edge on a scalar signal. This option is ignored for compound signals.

- -falling

  (optional) Specifies a search for falling edge on a scalar signal. This option is ignored for compound signals.

- -anyedge

  (optional) Specifies a search for a rising or falling edge on a scalar signal. This option is ignored for compound signals. (default)

- -startDelta <num>

  (optional) Indicates a simulation delta cycle on which to start.

  <num> — Any positive integer.

- -value <string>

  (optional) Specifies a match of a single scalar or compound signal against a specified string.

  <string> — Specifies a string to be matched.

- <startTime> [<unit>]

  (required) Specifies the simulation time at which to start the search. The time is specified as an integer or decimal number. Must be placed immediately before the <pattern> argument.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid VHDL time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- <pattern>

  (Required unless the -expr argument is used.) Specifies one or more signal names or wildcard patterns of signal names to search on. Must be specified as the final argument to the searchlog command.

Related Topics

- virtual signal
- virtual log
- virtual nolog
- GUI_expression_format

see

This command displays the specified number of source file lines around the current execution line and places a marker to indicate the current execution line. If specified without arguments, five lines will be displayed before and four lines after.

Syntax

see [<n> | <pre> <post>]

Arguments

- <n>

  (optional) Designates the number of lines to display before and after the current execution line.

- <pre>

  (optional) Designates the number of lines to display before the current execution line.

- <post>

  (optional) Designates the number of lines to display after the current execution line.

Examples

- Display 2 lines before and 5lines after the current execution line.

  see 2 5

  Returns:

```
#   92 :
#   93 :             if (verbose) $display("Read/Write test done");
# ->94 :             $stop(1);
#   95 :         end
#   96 :     end
#   97 :
#   98 :     or2 i1 (
#   99 :         .y(t_set),
```

seetime

This command scrolls the List or Wave window to make the specified time visible.

For the List window, a delta can be optionally specified as well.

Returns the specified time and unit.

Arguments to this command are order dependent. Please read the argument descriptions for more information. “<time> [<unit>]” must be specified as the final argument to the seetime command.

Syntax

seetime list | wave [-window <wname>] [-select] [-delta <num>] [-event] **<time> [<unit>]]**

Arguments

- list | wave

  (required) Controls which type of window to target. Must be specified as the first argument to the seetime command.

  list — Target a list window.

  wave — Target a wave window.

- -window <wname>

  (optional) Specifies that a window other than the current default Wave or List window is used. The default is to use the current default Wave or List window when the -window argument is not specified. Use the view command to change the default window.

  <wname> — Specifies an instance of the Wave or List window that is not the current default.

- -select

  (optional) Also moves the active cursor or marker to the specified time (and optionally, delta). Otherwise, the window is only scrolled.

- -delta <num>

  (optional) For the List window when deltas are not collapsed, this option specifies a delta. Otherwise, delta 0 is selected.

  <num> — Any positive integer.

- -event

  (optional) List or Wave window scrolls to the specified event.

- <time> [<unit>]

  <time> — (required) Specified as an integer or decimal number. Current simulation units are the default unless specifying <unit>. Must be specified as the final argument to the seetime command.

<unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid VHDL time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

setenv

This command changes or reports the current value of an environment variable. The setting is valid only for the current Questa SIM session.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

setenv <varname> [<value>]

Arguments

- <varname>

  (required) The name of the environment variable you wish to set or check. Must be specified as the first argument to the setenv command.

- <value>

  (optional) The new value for <varname>. If you do not specify a value, Questa SIM reports the variable's current value.

Related Topics

- unsetenv
- printenv

shift

This command shifts macro parameter values left one place, so that the value of parameter \$2 is assigned to parameter \$1, the value of parameter \$3 is assigned to \$2, and so on. The previous value of \$1 is discarded.

The shift command and macro parameters are used in macro files. If a macro file requires more than nine parameters, they can be accessed using the shift command.

To determine the current number of macro parameters, use the argc simulator state variable. Refer to "Simulator State Variables" for more information.

Syntax

shift

Arguments

None

Description

For a macro file containing nine macro parameters defined as $1 to $9, one shift command shifts all parameter values one place to the left. If more than nine parameters are named, the value of the tenth parameter becomes the value of $9 and can be accessed from within the macro file.

Related Topics

- do

show

This command lists HDL objects and subregions visible from the current environment.

The objects listed include:

- VHDL — signals, processes, constants, variables, and instances.
- Verilog — nets, registers, tasks, functions, instances, variables, and memories.

When using C Debug, show displays the names and types of the local variables and arguments of the current C function.

The show command returns formatted results to stdout. To eliminate formatting (to use the output in a Tcl script), use the Show form of the command instead.

Syntax

show [-all] [<pathname>]

Arguments

- -all

  (optional) Displays all names at and below the specified path recursively.

- <pathname>

  (optional) Specifies the pathname of the environment for which you want the objects and subregions to be listed; if omitted, the current environment is assumed.

Examples

- List the names of all the objects and subregion environments visible in the current environment.

  show

- List the names of all the objects and subregions visible in the environment named /uut.

  show /uut

- List the names of all the objects and subregions visible in the environment named sub_region which is directly visible in the current environment.

  show sub_region

- List the names of all the objects and subregions visible in all top-level environments.

  show -all /

Related Topics

- environment
- find

simstats

This command returns performance-related statistics about elaboration and simulation. The statistics measure the simulation kernel process (vsimk) for a single invocation of vsim. If you invoke vsim a second time, or restart the simulation, the current statistics are discarded and new values are collected.

If executed without arguments, the command returns a list of statistics and their related units in separate lines similar to the following:

```
# Memory Statistics
#     mem: size after elab (VSZ)            105348.00 Kb
#     mem: size during sim (VSZ)            171492.00 Kb
# Elaboration Time
#    elab: wall time                           0.41 s
#    elab: cpu time                            0.23 s
# Simulation Time
#     sim: wall time                           1.18 s
#     sim: cpu time                            0.66 s
# Tcl Command Time
#     cmd: wall time                         356.39 s
#     cmd: cpu time                            0.80 s
# Total Time
#   total: wall time                         357.98 s
#   total: cpu time                            1.68 s
```

The previous list output format can be obtained using the simstatslist command.

All statistics are measured at the time you invoke simstats. See the arguments below for descriptions of each statistic.

Units for time values are in seconds. Units for memory values are in Kbytes.

Different operating systems report these numbers differently.

Syntax

simstats [covsavecpu | covsavetime | elabcpu | elabmemory | elabtime | license | logcpu | logtime | simcpu | simmemory | simtime | tclcmdcpu | tclcmdtime | totalcpu | totaltime | verbose | voptcpu | voptmemory | vopttime]

Arguments

- covsavecpu

  (optional) Returns cpu time consumed by coverage save operations.

- covsavetime

  (optional) Returns wall clock time consumed by coverage save operations.

- elabcpu

  (optional) Returns cpu time consumed by vsim elaboration.

- elabmemory

  (optional) Returns memory consumed during vsim elaboration.

- elabtime

  (optional) Returns wall clock time consumed by vsim elaboration.

- license

  (optional) Returns a ‘License Statistics’ section that includes license statistics for checkout time and checked-out license feature names.

- logcpu

  (optional) Returns cpu time consumed by WLF logging.

- logtime

  (optional) Returns wall clock time consumed by WLF logging.

- simcpu

  (optional) Returns cumulative cpu time consumed by all run commands

- simmemory

  (optional) Returns memory consumed during the whole simulation, including the elaboration memory.

- simtime

  (optional) Returns cumulative wall clock time consumed by all run commands.

- tclcmdcpu

  (optional) Returns cpu time consumed by all TCL commands, excluding run commands.

- tclcmdtime

  (optional) Returns wall clock time consumed by all TCL commands, excluding run commands.

- totalcpu

  (optional) Returns total cpu time consumed by vsim command.

- totaltime

  (optional) Returns total wall clock time consumed by vsim command.

- verbose

  (optional) Displays verbose performance statistics, including an ‘elab’ report for checked-out license feature names.

- voptcpu

  (optional) Returns cpu time of vopt process. (Applicable in Two-Step Flow.)

- voptmemory

  (optional) Returns memory consumed by vopt process. (Applicable in Two-Step Flow.)

- vopttime

  (optional) Returns wall clock time of vopt process. (Applicable in Two-Step Flow.)

simstatslist

This command returns performance-related statistics about elaboration and simulation. Use this command in place of the simstats command to produce the original statistics output format as follows:

```
{{elab memory} 105348} {{sim memory} 171492} {{elab time} 0.410009} {{elab
cpu time} 0.234002} {{sim time} 1.18003} {{sim cpu time} 0.655204}
{{tclcmd time} 411.601} {{tclcmd cpu time} 0.795605} {{total time}
413.191} {{total cpu time} 1.68481}
```

All statistics are measured at the time you invoke simstatslist. See the arguments below for descriptions of each statistic.

Units for time values are in seconds. Units for memory values are in Kbytes.

Different operating systems report these numbers differently.

Syntax

simstatslist [covsavecpu | covsavetime | elabcpu | elabmemory | elabtime | logcpu | logtime | simcpu | simmemory | simtime | tclcmdcpu | tclcmdtime | totalcpu | totaltime | voptcpu | voptmemory | vopttime]

Arguments

- covsavecpu

  (optional) Returns cpu time consumed by coverage save operations

- covsavetime

  (optional) Returns wall clock time consumed by coverage save operations.

- elabcpu

  (optional) Returns cpu time consumed by vsim elaboration.

- elabmemory

  (optional) Returns memory consumed during vsim elaboration.

- elabtime

  (optional) Returns wall clock time consumed by vsim elaboration.

- logcpu

  (optional) Returns cpu time consumed by WLF logging.

- logtime

  (optional) Returns wall clock time consumed by WLF logging.

- simcpu

  (optional) Returns cumulative cpu time consumed by all run commands

- simmemory

  (optional) Returns memory consumed during the whole simulation, including the elaboration memory.

- simtime

  (optional) Returns cumulative wall clock time consumed by all run commands.

- tclcmdcpu

  (optional) Returns cpu time consumed by all TCL commands, excluding run commands.

- tclcmdtime

  (optional) Returns wall clock time consumed by all TCL commands, excluding run commands.

- totalcpu

  (optional) Returns total cpu time consumed by vsim command.

- totaltime

  (optional) Returns total wall clock time consumed by vsim command

- voptcpu

  (optional) Returns cpu time of vopt process. (Applicable in Two-Step Flow.)

- voptmemory

  (optional) Returns memory consumed by vopt process. (Applicable in Two-Step Flow.)

- vopttime

  (optional) Returns wall clock time of vopt process. (Applicable in Two-Step Flow.)

stack down

This command moves down the call stack.

If invoked without arguments, the command moves down the call stack by 1 level. The Locals window displays local variables at the level.

Syntax

stack down [n]

Arguments

- n

  (optional) Moves down the call stack by *n* levels. The default value is 1 level.

Related Topics

stack frame
stack level
stack tb
stack up

stack frame

This command selects the specified call frame.

Syntax

stack frame <n>

Arguments

- <n>

  Selects call frame number *n*. The currently executing frame is zero (also called the innermost) frame, frame one is the frame that called the innermost, and so on. The highest numbered frame is that of *main*.

Related Topics

stack down
stack level
stack tb
stack up

stack level

This command reports the current call frame number.

Syntax

stack level

Arguments

None

Related Topics

stack down
stack frame
stack tb
stack up

stack tb

This command displays a stack trace for the current process in the Transcript window. This lists the sequence of HDL function calls that have been entered to arrive at the current state for the active process. The stack tb command is an alias for the tb command.

If you are using C Debug, tb displays a stack trace of the C call stack.

Syntax

tb [<#_of_levels>]

Arguments

- <#_of_levels>

  (optional) Specifies the number of call frames in the C stack to display. If you do not specify a level, the entire C stack is displayed. This argument is available only for C Debug.

stack up

This command moves up the call stack.

If invoked without arguments, the command moves up the call stack by 1 level. The Locals window displays local variables at the level.

Syntax

stack up [n]

Arguments

- n

  (optional) Moves up the call stack by *n* levels. The default value is 1 level.

Related Topics

stack down
stack frame
stack level
stack tb

status

This command lists summary information about currently interrupted macros.

If invoked without arguments, the command lists the filename of each interrupted macro, the line number at which it was interrupted, and prints the command itself. It also displays any onbreak or onerror commands that have been defined for each interrupted macro.

Syntax

status [file | line]

Arguments

- file

  (optional) Reports the file pathname of the current macro.

- line

  (optional) Reports the line number of the current macro.

Examples

The transcript below contains examples of resume, and status commands.

```
VSIM(paused)> status
# Macro resume_test.do at line 3 (Current macro)
#     command executing: "pause"
#     is Interrupted
#     ONBREAK commands: "resume"
# Macro startup.do at line 34
#     command executing: "run 1000"
#     processing BREAKPOINT
#     is Interrupted
#     ONBREAK commands: "resume"
VSIM(paused)> resume
# Resuming execution of macro resume_test.do at line 4
```

Related Topics

- abort
- do
- pause
- resume

step

The step command is an alias for the run command with the -step switch. Steps the simulator to the next HDL or C statement.

Current values of local HDL variables may be observed at this time using the Locals window.

Syntax

step [-current] [<n>] [-out] [-over [<n>]] [-this "this==<class_handle>"]

Arguments

- -current

  (optional) Instructs the simulation to step into an instance, process, or thread and stay in the current thread. Prevents stepping into a different thread.

- <n>

  Moves the simulator <n> steps ahead. Moves the debugger <n> lines ahead when you are using C Debug. Specified as a positive integer value.

- -out

  (optional) Instructs the simulation to step out of the current function or procedure and return to the caller.

- -over

  (optional) Directs Questa SIM to run VHDL procedures and functions, Verilog tasks and functions, and C functions but to treat them as simple statements instead of entering and tracing them line by line.

  You can use the -over argument to skip over a VHDL procedure or function, Verilog task or function, or a C function.

  When a wait statement or end of process is encountered, time advances to the next scheduled activity. Questa SIM then updates the Process and Source windows to reflect the next activity.

- -this "this==<class_handle>"

  (optional) Instructs the simulation to step into a method of a SystemVerilog class when “this” refers to the specified class handle. To obtain the handle of the class, use the examine -handle command.

  <class_handle> — Specifies a SystemVerilog class. Note that you must use quotation marks (" ") with this argument.

Description

You can control any return values after the step operation completes with the following preference variables:

- noRunTimeMsg — Set this variable to 0 to display simulation time and delta information or set it to 1 to disable the display of this information.
- noRunStatusMsg — Set this variable to 0 to display run status informatin or set it to 1 to disable the display of this information.

The following is an example that shows a series of run commands (the step command behaves similarly) and how the output changes with the preference variable settings:

```
VSIM 1> run 105
VSIM 2> set PrefMain(noRunTimeMsg) 0
# 0

VSIM 3> run 112
# Time: @217 ns 0

VSIM 4> set PrefMain(noRunStatusMsg) 0
# 0

VSIM 5> run 100
# Time: @317 ns 0
# Status: ready end

VSIM 6> set PrefMain(noRunTimeMsg) 1
# 1

VSIM 7> run 50
# Status: ready end

VSIM 8> set PrefMain(noRunStatusMsg) 1
# 1

VSIM 9> run 55

VSIM 10>
```

Related Topics

- run
- Stepping Through Your Design

stop

This command is used with the when command to stop simulation in batch files.

The stop command has the same effect as hitting a breakpoint. The stop command may be placed anywhere within the body of the when command.

Syntax

stop [-sync]

Arguments

- -sync

  (optional) Stops the currently running simulation at the next time step.

Description

Use run -continue to continue the simulation run, or the resume command to continue macro execution. If you want macro execution to resume automatically, put the resume command at the top of your macro file:

```
onbreak {resume}
```

Note

If you want to stop the simulation using a when command, you must use a stop command within your when statement. DO NOT use an exit command or a quit command. The stop command acts like a breakpoint at the time it is evaluated.

Related Topics

- bp
- resume
- run
- when

suppress

This command prevents one or more specified messages from displaying. You cannot suppress Fatal or Internal messages. The suppress command used without arguments returns the message numbers of all suppressed messages.

Note

To use the suppress command, you must have a design loaded in Questa SIM. Otherwise, Questa SIM will display an error message without running the command.

Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

Syntax

suppress [-clear <msg_number>[,<msg_number>,...]] [<msg_number>[,<msg_number>,...]] [<code_string>[, <code_string>,...]]

Arguments

- -clear <msg_number>[,<msg_number>,...]

  (optional) Clears suppression of one or more messages identified by message number.

  <msg_number> — A number identifying the message. Multiple message numbers are specified as a comma separated list.

- <msg_number>[,<msg_number>,...]

  (optional) A number identifying the message to be suppressed. Multiple message numbers are specified as a comma separated list.

- <code_string>[, <code_string>,...]

  (optional) A string identifier of the message to be suppressed. Disables warning messages in the category specified by <code_string>. Warnings that can be disabled include the <code_string> name in square brackets in the warning message.

Examples

- Return the message numbers of all suppressed messages:

  suppress

- Suppress messages by message number:

  suppress 8241,8242,8243,8446,8447

- Suppress messages by numbers and code categories:

  suppress 8241,TFMPC,CNNODP,8446,8447

- Clear message suppression for the designated messages:

  suppress -clear 8241,8242

- Return the message numbers of all suppressed messages and clear suppression for all:

  suppress -clear [suppress]

tb

This (traceback) command (traceback) displays a stack trace for the current process in the Transcript window. This lists the sequence of HDL function calls that have been entered to arrive at the current state for the active process.

If you are using C Debug, tb displays a stack trace of the C call stack.

Syntax

tb [<#_of_levels>]

Arguments

- <#_of_levels>

  (optional) Specifies the number of call frames in the C stack to display. If you do not specify a level, the entire C stack is displayed. This argument is available only for C Debug.

tcheck_set

This command works in tandem with tcheck_status to report on and enable/disable individual timing checks. tcheck_set modifies either a check's reporting or X-generation status, and reports the new setting in the Transcript window.

Disabling a timing check's reporting prevents generation of associated violation messages. For Verilog modules this means Questa SIM disables message reporting. For VHDL design units this means Questa SIM sets the MsgOn parameter in a VITAL timing check procedure (TCP) to FALSE. Disabling a timing check's X generation removes a timing check's ability to affect the outputs of the simulation. For Verilog modules, this means Questa SIM toggles the timing check's notifier. For VHDL design units, this means Questa SIM sets the Xon parameter in a VITAL TCP to FALSE.

Note that the tcheck_set command *does not* override the effects of invoking vlog or vsim with the +nospecify, +notimingchecks, or +no_neg_tchk arguments. Using tcheck_set can override the effects of invoking vsim with the +no_notifier, +no_tchk_msg, -g, or -G arguments. These latter arguments establish initial values for the simulation, and those values can be modified by tcheck_set.

Keep in mind the following if you are using VHDL VITAL:

- VITAL does not provide the granularity to set individual period or width checks. These checks are part of a single VITAL TCP, and tcheck_set toggles MsgOn and Xon for all checks in the TCP. See "Examples" below for further information.
- If an instance is not Level-1 optimized, you cannot set values for individual TCPs. You can set values only for the entire instance. tcheck_status reports "ALL" for instances that aren't Level-1 optimized. See "Examples" below for further information.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

tcheck_set [-quiet] **<instance>** [{-m | -n}] [-r [-v]] [<tcheck> | ALL]
[<Stat> | <MsgStat> <XStat>]

Arguments

- <instance>

  (required) Specifies the instance for which you want to change the reporting or X-generation status.

- -m | -n

  (optional) Specifies whether *instance* is a module/entity or a net.

  -m — Indicates that the *instance* is a module or entity and that tcheck_set will operate on all instances of the specified module type.

-n — Indicates that the *instance* is a hierarchical pathname to a net and that tcheck_set will operate on all instances connected to the specified net.

- -quiet

  (optional) Suppresses printing the new setting to the Transcript window. If you use this switch, it must precede the *instance* argument. You can specify -q as an alias of this switch.

- -r [-v]

  (optional) Attempts to change all checks on *instance* and all instances below (recursive). By default, the recursively altered instances are not printed to the transcript.

  -v — (optional) Specifies that recursively altered instances should be printed to the transcript.

- <tcheck> | ALL

  (optional) Specifies which checks should be changed.

  <tcheck> — (optional) Specifies a specific timing check to change. You can specify either:

  - The integer that is assigned to each timing check (and reported via tcheck_status). Note that the integer number may change between library compiles.
  - The full timing check name enclosed in double quotes, such as:

    “(WIDTH (negedge CLK))”
  - The type of timing check, with out any port identifiers, enclosed in double quotes. This acts as a form of wildcard specification, such as:

    “(WIDTH)”

    which instructs tcheck_set to operate on all timing checks with the type “WIDTH” in *instance*.

  ALL — Attempts to change all checks in *instance*. (default)

- <Stat>

  (optional) Enables/disables both violation message and X generation reporting for the specified timing check(s).

  ON — Enable both violation and X generation reporting.

  OFF — Disable both violation and X generation reporting.

- <MsgStat> <XStat>

  (optional) Individually controls both violation message and X generation reporting for the specified timing check(s). Both variables, <MsgStat> and <XStat>, are specified together. <MsgStat> must be specified before <XStat>. For example,

  ON OFF— Enable violation message reporting and disable X generation reporting.

  OFF ON — Disable violation message reporting and enable X generation reporting.

Examples

- Turn off message reporting and X generation for the "(WIDTH (negedge CLK))" check in instance *top.y1.u2*. These examples assume that your PathSeparator variable is set to "." rather than the default "/".

  tcheck_set top.y1.u2 "(WIDTH (negedge CLK))" OFF

  Returns the following output in the Transcript window:

  ```
  #0 ( WIDTH (negedge  CLK) ) MsgOff XOff
  ```

- Turn off message reporting for timing check number 1 in instance *top.y1.u2*.

  tcheck_set top.y1.u2 1 OFF ON

  Returns the following output in the Transcript window:

  ```
  #1 ( WIDTH (posedge  CLK) ) MsgOff XOn
  VSIM 2> tcheck_status dff1
  # 1 ( PERIOD CLK ) MsgOn, XOn
  #     ( WIDTH (posedge CLK) ) MsgOn, XOn
  #     ( WIDTH (negedge CLK) ) MsgOn, XOn
  VSIM 3> tcheck_set dff1 "( WIDTH (posedge CLK) )" off on
  # 1 ( PERIOD CLK ) MsgOff, XOn
  #     ( WIDTH (posedge CLK) ) MsgOff, XOn
  #     ( WIDTH (negedge CLK) ) MsgOff, XOn
  ```

  Show how period and hold checks work with VHDL VITAL. In this case, specifying "off on" for (WIDTH (posedge CLK)) also sets (PERIOD CLK) and (WIDTH (negedge CLK)) to the same values.

  ```
  VSIM 3> tcheck_status dff5
  # ALL MsgOn XOn
  VSIM 4> tcheck_set dff5 on off
  # ALL MsgOn XOff
  ```

  Instance *dff5* is from an unaccelerated model so tcheck_set can only toggle message reporting and X generation for all checks on the instance.

- Turn off message reporting and X generation on all WIDTH statements on modules of type "mymod".

  tcheck_set mymod -m "(WIDTH)" OFF

- Recursively turn on message reporting and turn off X generation on all SETUP statements on instances driven by the net top.clk.

  tcheck_set top.clk -r -n "(SETUP)" ON OFF

Related Topics

- tcheck_status
- Compiling and Simulating with Accelerated VITAL Packages
- SDF Timing Annotation
- Disabling Timing Checks
- vsim -g
- vsim -G
- vsim +no_notifier
- vsim +no_tchk_msg
- vsim +nospecify
- vsim +no_neg_tchk
- vsim +notimingchecks

tcheck_status

This command works in tandem with tcheck_set to report on and enable/disable individual timing checks. tcheck_status prints in the Transcript window the current status of all timing checks in the instance or a specific timing check specified with the optional <tcheck> argument.

You can prevent the generation of associated violation messages by disabling the reporting of a timing check. For Verilog modules this means Questa SIM disables message reporting. For VHDL design units this means Questa SIM sets the MsgOn parameter in a VITAL timing check procedure (TCP) to FALSE. Disabling a timing check's X generation removes a timing check's ability to affect the outputs of the simulation. For Verilog modules this means Questa SIM toggles the timing check's notifier. For VHDL design units this means Questa SIM sets the Xon parameter in a VITAL TCP to FALSE.

Syntax

tcheck_status [-lines] [-m | -n] [-r] **<instance>** [<tcheck>]

You must specify the arguments in the order shown.

Arguments

- <instance>

  (required) Specifies the instance for which you want timing check status reported.

- -m | -n

  (optional) Specifies whether *instance* is a module/entity or a net.

  -m — Indicates that the *instance* is a module or entity and that tcheck_status will operate on all instances of the specified module type.

  -n — Indicates that the *instance* is a hierarchical pathname to a net and that tcheck_status will operate on all instances connected to the specified net.

- -lines

  (optional) Specifies that the HDL source file and line numbers of the check(s) be displayed. Has no effect on VHDL instances. Note that line information may not always be available. You can specify -l as an alias for this switch.

- -r

  (optional) Operates recursively on all instances.

- <tcheck>

  (optional) Specifies a specific timing check to report. You can specify either:

  - The integer that is assigned to each timing check (and reported via tcheck_status). Note that the integer number may change between library compiles.
  - The full timing check name enclosed in double quotes, such as:

    "(WIDTH (negedge CLK))"

- The type of timing check, with out any port identifiers, enclosed in double quotes. This acts as a form of wildcard specification, such as:

  "(WIDTH)"

  which instructs tcheck_set to operate on all timing checks with the type "WIDTH" in *instance*.

Return Values

The output of tcheck_status is displayed in the following form:

```
#<Number> <SDF_Description> [<src_line>] <MsgStat> <XStat>
```

Table 2-9 contains a short description of each field in the output.

Table 2-9. Output Fields for tcheck_status Command

Field	Description
<Number>	an integer that can be used as shorthand to specify the check in the tcheck_status or tcheck_set commands (as the <tcheck> argument); this number can change with compiler optimizations, and you can't assume it will stay the same between library compiles
<SDF_Description>	an SDF specification of the timing check including enclosing parentheses '( )'
<src_line>	the source file and line number for the timing check specification; output if you specify the -lines argument; the format of the object is <source_file_name>:<line_number>.
<MsgStat>	violation message reporting status indicator • MsgON/MsgOFF — violation reporting is enabled/disabled and unchangeable • MsgOn/MsgOff — violation reporting is enabled/disabled and modifiable
<XStat>	violation X generation status indicator • XON/XOFF — X generation is enabled/disabled and unchangeable • XOn/XOff — X generation is enabled/disabled and modifiable

Examples

- Report the timing checks for the instance "top.y1.u2":

  tcheck_status top.y1.u2

  Returns:

```
#0 ( WIDTH (negedge  CLK) ) MsgOn XOn
#1 ( WIDTH (posedge  CLK) ) MsgOn XOn
#2 ( SETUP (negedge  D) (posedge  CLK) ) MsgOFF XOFF
#3 ( HOLD (posedge  CLK) (negedge  D) ) MsgOn XOff
```

- Report the timing checks for the instance “top.y1.u2” with line numbers:

  tcheck_status -lines top.y1.u2 1

  Returns:

  ```
  #1 ( WIDTH (posedge  CLK) ) 'cell.v:224' MsgOn XOn
  ```

- Report the timing check with the description “WIDTH (posedge CLK)”:

  tcheck_status -lines top.y1.u2 "WIDTH (posedge CLK)"

  Returns:

  ```
  #1 ( WIDTH (posedge  CLK) ) 'cell.v:224' MsgOn XOn
  ```

- Report the timing check associated with the tcheck number “1”:

  tcheck_status -lines top.y1.u2 1

  Returns:

  ```
  #1 ( WIDTH (posedge  CLK) ) 'cell.v:224' MsgOn XOn
  ```

- Report the status of all WIDTH statements on modules of type "mymod".

  tcheck_status mymod -m "(WIDTH)"

- Recursively report the status of all SETUP statements on instances driven by the net top.clk.

  tcheck_status top.clk -r -n "(SETUP)"

Related Topics

- tcheck_set
- SDF Timing Annotation

Time

These commands allow you to perform comparisons between, operations on, and conversions of, time values.

When [unit] is left unspecified the commands assume the current simulation time unit, as defined by the Resolution variable in the *modelsim.ini* file or the vsim -t command. Units of time (ns, us, and so forth) can be specified independently for each <time[1 | 2]> specified for most of the commands. See the description of each command and examples for more information.

Arguments to this command are order dependent. Follow the order specified in the Syntax for each command.

Syntax

eqTime **<time1>[unit] <time2>[unit]**

Returns a 1 (true) or 0 (false) if <time1> and <time2> are equal.

neqTime **<time1>[unit] <time2>[unit]**

Returns a 1 (true) or 0 (false) if <time1> and <time2> are not equal.

ltTime **<time1>[unit] <time2>[unit]**

Returns a 1 (true) or 0 (false) if <time1> is less than <time2>.

gtTime **<time1>[unit] <time2>[unit]**

Returns a 1 (true) or 0 (false) if <time1> is greater than <time2>.

lteTime **<time1>[unit] <time2>[unit]**

Returns a 1 (true) or 0 (false) if <time1> is less than or equal to <time2>.

gteTime **<time1>[unit] <time2>[unit]**

Returns a 1 (true) or 0 (false) if <time1> is greater than or equal to <time2>.

addTime **<time1>[unit] <time2>[unit]**

Returns the sum of adding <time1> to <time2>

subTime **<time1>[unit] <time2>[unit]**

Returns the value of subtracting <time2> from <time1>

mulTime **<time1>[unit] <integer>**

Returns the value of multiplying <time1> by an <integer>

divTime **<time1>[unit] <time2>[unit]**

Returns an integer, that is the value of dividing <time1> by <time2>. Specifying 0 for <time2> results in an error.

intToTime **<high_32bit_int> <low_32bit_int>**

Returns a 64-bit time value based on two 32-bit parts of a 64-bit integer. This command is useful when you've performed an integer calculation that results in a 64-bit value and need to convert it to a time unit.

scaleTime **<time1>[unit] <scale_factor>**

Returns a time value scaled by a real number and truncated to the current time resolution.

RealToTime **<real>**

Returns a time value equivalent to the specified real number and truncated to the current time resolution.

validTime **<time>**

Returns a 1 (true) or 0 (false) if the given string is a valid time for use with any of these Time calculations.

formatTime **{+ | -} commas | {+ | -}nodefunit | {+ | -}bestunits**

Sets display properties for time values.

Arguments

- <time1>[unit]

  <time> — Specified as an integer or decimal number. Current simulation units are the default unless specifying <unit>.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid VHDL time units are:

  fs — femtosecond (10^{-15} seconds)

  ps — picosecond (10^{-12} seconds)

  ns — nanosecond (10^{-9} seconds)

  us — microsecond (10^{-6} seconds)

  ms — millisecond (10^{-3} seconds)

  sec — second

  min — minute (60 seconds)

  hr — hour (3600 seconds)

  Note that if you put a space between the values, you must enclose the argument in braces ({ }) or quotation marks (" ").

- <high_32bit_int> | <low_32bit_int>

  <high_32bit_int> — The "high" part of the 64-bit integer.

  <low_32bit_int> — The "low" part of the 64-bit integer.

- <scale_factor> — a real number to be used as scaling factor. Common values can include:

```
0.25, 0.5, 1.5, 2, 10, 100
```

- {+ | -} commas — controls whether commas are displayed in time values.

  +commas — time values include commas

  -commas — time values do not include commas

- {+ | -}nodefunit — controls whether time values display time units

  +nodefunit — time values do not include time units and will be in current time resolution

  -nodefunit — time values may include time units

- {+ | -}bestunits — controls whether time values display the largest possible time unit, for example 8 us instead of 8,000 ns.

  +bestunits — time values display the largest possible time unit

  -bestunits — time values display the default time unit

Examples

>ltTime 100ns 1ms

Returns:

```
# 1
```

>addTime {1545 ns} {455 ns}

Returns:

```
# 2 us
```

>gteTime "1000 ns" "1 us"

Returns:

```
# 1
```

>divTime 1us 10ns

Returns:

```
# 100
```

>formatTime +bestunit

Returns:

```
# -commas -nodefunit +bestunits
```

>scaleTime 3ms 1000

Returns:

```
# 3 sec
```

>RealToTime 1.345e04

Returns:

```
# 13450 ns
```

toggle add

The toggle add command enables collection of toggle statistics for the specified nodes. Arguments specified with toggle add command override those set by the +cover=t(x) arguments set at compile time (vlog/vcom). For example, if toggle add -full is given for signals already added as standard (two transition) toggles, they will be converted to extended toggle coverage mode (all six transitions). Similarly, if toggle add command is used on extended toggle coverage mode toggles (six transitions), they will be converted into standard coverage toggles (two transitions).

The allowed nodes are Verilog nets and registers and VHDL signals of type bit, bit_vector, std_ulogic, std_logic, and std_logic_vector. Also, VHDL Boolean and Integer types (including subranges) and other user-defined Enum types, as well as SystemVerilog real types, are supported for use. All other types are silently ignored.

You may change toggle mode (from basic, t, to extended, x, or vice versa) during simulation. 0L->1H and 1H-> 0L transition counts are maintained no matter how many times you switch between basic/extended toggle modes. Z transitions, however, are reset. See Using the Toggle Add Command.)

You can also collect and view toggle statistics in the Questa SIM GUI. Refer to “Coverage” for details.

Syntax

toggle add [-exclude {<list>}] [-countlimit <int>] [-extendedtogglemode 1|2|3] [-full] [-in] [-inout] [-internal] [-out] [-ports] [-r] [-unique] [-widthlimit] <node_name>

Arguments

- -exclude {<list>}

  (optional) Excludes specified bits of a bus from toggle computations and reports. Can also be used to exclude specific VHDL or SystemVerilog enums or ranges of enums from toggle coverage and reporting.

  <list> is a space-separated list of integers or ranges, where a range is two integers separated by either ":" or "-". The range must be in the same ascending or descending order as the signal declaration. If a second toggle add -exclude command is issued on the same signal, the latest command will override the earlier one.

- -countlimit <int>

  (optional) Limits the toggle coverage count for a toggle node. Overrides the global default value of <int> (which is ‘1’), set by the ToggleCountLimit *modelsim.ini* variable.

- -extendedtogglemode 1|2|3

  (optional) Sets the desired level for extended toggles. Coverage calculation criteria for these three levels of support are:

  1 — 0L->1H & 1H->0L & any one 'Z' transition (to/from 'Z')

  2 — 0L->1H & 1H->0L & one transition to 'Z' & one transition from 'Z'

3 — 0L->1H & 1H->0L & all 'Z' transitions

- -full

  (optional) Enables extended mode toggle coverage for the specified toggles, which tracks the following six transitions:

  - 1 or H --> 0 or L
  - 0 or L --> 1 or H
  - Z --> 1 or H
  - Z --> 0 or L
  - 1 or H --> Z
  - 0 or L --> Z

  By default (normal, non-extended mode) only the first two transitions – to and from 0 and to and from1 are counted. When normal two state toggle is converted to extended toggle during run-time, 0->1 and 1->0 counts are preserved, however all Z transition counts start from zero.

  Performing a toggle add command on a group of signals and converting to extended toggle coverage mode (all six transitions) using toggle add -full on the same signals converts the signals to extended mode. Similarly, a toggle add command performed on signals already present in extended toggle coverage mode converts the signals to a two state mode.

  For SystemVerilog reg-vector toggles, only whole signals can be converted to or from extended mode, not the individual bits.

  For SystemVerilog struct, conversion applies to all fields of the structure, not solely to a specified type of field.

- -in

  (optional) Enables toggle statistics collection on nodes of mode IN.

- -inout

  (optional) Enables toggle statistics collection on nodes of mode INOUT.

- -internal

  (optional) Enables toggle statistics collection on internal (non-port) objects.

- -out

  (optional) Enables toggle statistics collection on nodes of mode OUT.

- -ports

  (optional) Enables toggle statistics collection on nodes of modes IN, OUT, or INOUT.

- -r

  (optional) Specifies that toggle statistics collection is enabled recursively into subregions. If omitted, toggle statistic collection is limited to the current region.

- -unique

  (optional) Reports an attempt to add a signal that is an alias to a signal already added. The alias will not be added.

- -widthlimit

  (optional) Limits the maximum width of signals included in toggle coverage for the specified node. Overrides the global default limit (128) set by the ToggleCountLimit *modelsim.ini* variable.

- <node_name>

  (required) Enables toggle statistics collection for the named node(s). Multiple names and wildcards are accepted.

Returns

Command result	Return value
no signals are added, no signals are found to be already in the toggle set, and no existing toggle modified	Nothing added.
no signals are added and some signals are found to be already in the toggle set; but all mentioned existing toggles converted to standard/extended mode	0
some signals are added and all mentioned existing toggles converted to standard/extended mode	the number of bits added

Examples

- Enable toggle statistics collection for signal */dut/data/a*.

```
toggle add /dut/data/a
```

- Enable toggle statistics collection for bit 6 of bus */dut/data_in*. The curly braces must be added in order to escape the square brackets ('[]')

```
toggle add {/dut/data_in[5]}
```

Related Topics

- "Toggle Coverage"
- "Toggle Exclusion Management"
- toggle disable
- toggle enable
- toggle report
- toggle reset

toggle disable

The toggle disable command disables toggle coverage statistics collection on the specified nodes. The command provides a method of implementing coverage exclusions for toggle coverage. An equivalent command for excluding toggles from coverage is “coverage exclude -togglenode”.

For information on disabling individual bits, see Exclude Nodes from Toggle Coverage in the User’s Manual.

Syntax

toggle disable [-all] | [-in] [-out] [-inout] [-internal] [-ports] [-r] <node_name>

Arguments

- -all

  (optional) Disables toggle statistics collection for all nodes that have toggle checking enabled. Must be used alone without other arguments.

- -in

  (optional) Disables toggle statistics collection on nodes of mode IN.

- -out

  (optional) Disables toggle statistics collection on nodes of mode OUT.

- -inout

  (optional) Disables toggle statistics collection on nodes of mode INOUT.

- -internal

  (optional) Disables toggle statistics collection on internal (non-port) objects.

- -ports

  (optional) Disables toggle statistics collection on nodes of modes IN, OUT, or INOUT.

- -r

  (optional) Specifies that toggle statistics collection is disabled recursively into subregions. If omitted, the disable is limited to the current region.

- <node_name>

  (required) Disables toggle statistics collection for the named node(s). Multiple names and wildcards are accepted.

Related Topics

- coverage exclude
- "Toggle Coverage"
- toggle add
- toggle enable
- toggle report
- toggle reset

toggle enable

The toggle enable command re-enables toggle statistics collection on nodes whose toggle coverage had previously been disabled.

Syntax

toggle enable [-all] | [-in] [-out] [-inout] [-internal] [-ports] [-r] <node_name>

Arguments

- -all

  (optional) Enables toggle statistics collection for all nodes that have toggle checking disabled. Must be used alone without other arguments.

- -in

  (optional) Enables toggle statistics collection on disabled nodes of mode IN.

- -out

  (optional) Enables toggle statistics collection on disabled nodes of mode OUT.

- -inout

  (optional) Enables toggle statistics collection on disabled nodes of mode INOUT.

- -internal

  (optional) Enables toggle statistics collection on disabled internal (non-port) objects.

- -ports

  (optional) Enables toggle statistics collection on disabled nodes of modes IN, OUT, or INOUT.

- -r

  (optional) Specifies that toggle statistics collection is enabled recursively into subregions. If omitted, the enable is limited to the current region.

- <node_name>

  (required) Enables toggle statistics collection for the named node(s). Multiple names and wildcards are accepted.

Related Topics

- coverage exclude
- "Toggle Coverage"
- "Toggle Exclusion Management"
- toggle add
- toggle disable
- toggle report
- toggle reset

toggle report

The toggle report command displays a list of all unique nodes that have not transitioned to both 0 and 1 at least once, and the counts for how many times each node toggled for each state transition type.

Also displayed is a summary of the number of nodes checked, the number that toggled, the number that didn't toggle, and a percentage that toggled.

You can also collect and view toggle statistics in the Questa SIM GUI. Refer to "Coverage" for details.

The toggle report command is intended to be used as follows:

1. Enable statistics collection with the toggle add command.
2. Run the simulation with the **run** command.
3. Produce the report with the toggle report command.

Note

To ensure that you are reporting all signals in the design, use the toggle report -duplicates switch.

Ordering of toggle nodes

The ordering of nodes in the report may vary depending on how you specify the signal list. If you use a wildcard in the signal argument (e.g., toggle report -all -r /*), the nodes are listed in the order signals are found when searching down the context tree using the wildcard. Multiple elements of the same net will be listed multiple times. If you do not use a wildcard (e.g., toggle report -all -r /*), the nodes are listed in the order in which they were originally added to toggle coverage, and elements are not duplicated.

Syntax

toggle report [-all] [-duplicates] [-file <filename>]
[-select {inputs | outputs | inout | ports | internals}]
[-instance <path> [-recursive]] [-onexit] [<signal>…] [-showambiguity] [-summary] [-top]
[-verbose]

Arguments

- -all

  (optional) Lists all nodes, both toggled and untoggled. Does not list alias nodes.

- -duplicates

  (optional) Reports all the nodes, including alias nodes, using their local names. Alias nodes are marked with a '+' sign indicator, and do not affect coverage numbers.

- -file <filename>

  (optional) Specifies a file to which to write the report. By default the report is displayed in the Transcript window.

- -instance <path> [-recursive]

  (optional) Reports on toggles for a specified instance, and optionally on toggles under the specified instance path.

  The optional -recursive argument specifies that toggle statistics reporting is enabled recursively into subregions. If omitted, toggle statistic reporting is limited to the current region.

- -onexit

  (optional) Causes Questa SIM to report toggle data automatically when the simulator exits.

- <signal>…

  (optional) Specifies the name of a signal whose toggle statistics are to be displayed. Multiple signal names, separated by spaces, may be specified. Wildcards may be used.

- -select {inputs | outputs | inout | ports | internals}

  (optional) Reports on input, output, inout, all ports, or internal signals.

- -showambiguity

  (optional) When used, toggle report displays both minimum and maximum counts for any conflicting toggle data in a UCDB that results from a combined merge (vcover merge -combine).

- -summary

  (optional) Selects only the summary portion of the report.

- -top

  (optional) For signals that were added to toggle coverage using vcom or vlog -cover t, -top uses the name of the top-most element of multiple-segment (collapsed) nets. By default, the name of the wildcard-matching segment is used.

- -unique

  This option is obsolete with version 6.3. By default, toggles are always unique.

- -verbose

  (optional) Specifies that the toggle report includes all values taken on by integer variables.

Related Topics

- "Toggle Coverage"
- toggle add
- toggle disable
- toggle enable
- toggle reset

toggle reset

The toggle reset command resets the toggle counts to zero for the specified nodes.

Syntax

toggle reset [-all] | [-in] [-out] [-inout] [-internal] [-ports] [-r] <node_name>

Arguments

- -all

  (optional) Resets toggle statistics collection for all nodes that have toggle checking enabled. Must be used alone without other arguments.

- -in

  (optional) Resets toggle statistics collection on nodes of mode IN.

- -out

  (optional) Resets toggle statistics collection on nodes of mode OUT.

- -inout

  (optional) Resets toggle statistics collection on nodes of mode INOUT.

- -internal

  (optional) Resets toggle statistics collection on internal (non-port) objects.

- -ports

  (optional) Resets toggle statistics collection on nodes of modes IN, OUT, or INOUT.

- -r

  (optional) Specifies that toggle statistics collection is reset recursively into subregions. If omitted, the reset is limited to the current region.

- <node_name>

  (required) Resets toggle statistics collection for the named node(s). Multiple names and wildcards are accepted.

Related Topics

- "Toggle Coverage"
- toggle add
- toggle disable
- toggle enable
- toggle report

tr color

This command changes the color scheme of individual transactions and entire streams, either in a specific wave window or for all wave windows. It is the command equivalent of the Colors tab in the Transaction-Stream Properties dialog.

<color> in the arguments specifies the color to use. Either a standard X Window color name or an RGB value (e.g., #357f77) is accepted; multi-word names ("light blue") must be enclosed in quotes.

If no arguments are specified, this command returns the value of each configuration item in a Tcl list.

Unique abbreviations are accepted for all arguments.

Syntax

tr color {**-stream <stream> [<stream>] ...** | **-transaction <uid> [<uid>] ...**} [-attrbg <color>] [-attrtext <color>] [-border <color>] [-color <color>] [-default] [-get] [-inactive <color>] [-namebg <color>] [-nametext <color>] [-win <wave>]

Arguments

- -stream <stream> [<stream>] ...

  (required unless specifying -transaction.) If present, all objects specified in the tr color command are transaction streams. <stream> is the path to a transaction stream. Multiple streams are allowed, and the <stream> need not immediately follow the -stream argument. No wildcards are allowed.

- -transaction <uid> [<uid>] ...

  (required unless specifying -stream.) Specifies the Unique IDs (UID) of the transactions to configure. The UID consists of dataset name and the 64-bit serial number assigned during simulation, which can be determined using the "tr uid" command.

  <uid> can be specified either with full UID or just the serial number. If only the serial number is present, the current dataset as returned by the "env" command is assumed. If the full UID is used, it must be surrounded by curly braces ({}).

  Multiple ID specifications are allowed, and the <uid> need not immediately follow the -transaction argument. No wildcards are allowed.

- -attrbg <color>

  (optional) Select the color to use as the background for all attributes. See <color>.

- -attrtext <color>

  (optional) Select the color to use for attribute text.

- -border <color>

  (optional) Select the border color.

- -color <color>

  (optional) Select the background color for the transaction. All other colors are chosen automatically.

- <color>

  Specifies the color to use. Either a standard X Window color name or an RGB value (e.g., #357f77) is accepted; multi-word names (“light blue”) in quotes.

- -default

  (optional) Removes any color overrides on the specified streams or transactions. If present, this option takes precedence over any other option that sets color.

- -get

  (optional) Indicates that the command should return a list of the color schemes for each transaction or stream. If colors are changed by the command, this argument returns the resulting color scheme. Each scheme is itself a Tcl list with the colors listed in the following order: inactive line, border line, name background, name text, attribute background, attribute text.

- -inactive <color>

  (optional) Select the inactive-line color.

- -namebg <color>

  (optional) Select the color to use as the background for the transaction’s name.

- -nametext <color>

  (optional) Select the color for the transaction name.

- -win <wave>

  (optional) Specifies the wave window for which the changes should apply. <wave> is the Tk name (not the title) for the wave window. Any color changes to specific transactions take precedence over color changes to the streams carrying those transactions. You can change the scheme for the associated streams and not change those transactions. To remove color changes on specific transactions, use the –default option. The selected transactions would then reflect the color scheme of the stream.

Examples

- Set colors for the name and background for a specified transaction stream:

  tr color -stream -namebg "light blue" -nametext black /path/tr03

- Set the color of the border for a specified transaction:

  tr color -transaction -border #357f77 {sim 10023}

Related Topics

- Recording and Viewing Transactions
- tr uid
- tr order

tr order

This command controls which attributes are visible and the order in which they appear. It applies to entire streams only, either in a specific wave window or for all wave windows. It is the command equivalent of the Order tab in the Transaction-Stream Properties dialog.

This command functions to either:

- specify which attributes are visible and the order of those attributes (using -attributes and -default)
- display the attribute order and visibility settings (using -hidden and -visible).

Because two streams may have different attributes or the same attributes in different order, this command resolves the differences when setting the attribute order and visibility. When you set the order with -attributes, only attributes applying to a specific stream are visible. All other attributes for that stream are hidden. Names not matching actual attributes are ignored for that stream.

When restoring the original order with -default, each stream returns to its original order and visibility which may be different from that of another stream in the command line.

Unique abbreviations are accepted for all arguments.

Syntax

tr order [[-attributes <attrs>] [-default] [-win <wave>] **<stream> ...**] |

[-hidden] [-visible] [-win <wave>] {**<stream> ...**}]

Arguments

- -attributes <attrs>

  (optional) Specifies that the attributes for the specified stream(s) should be visible. All other attributes are hidden. The order of the list determines the order of the attributes listed. <attrs> is a Tcl list of attribute names; if more than one attribute is listed, the list must be surrounded by curly braces “{}”. Use an empty list of attributes (“{ }” to specify that there should be no visible attributes.

- -default

  (optional) Removes any visibility and order overrides on the specified streams or transactions. If present, this option takes precedence over the –attributes option.

- -hidden

  (optional) Return a list of the hidden attributes for each stream or transaction specified. If -visible is set, the hidden attributes are in a list following the visible attributes.

- -visible

  (optional) Return a list of the visible attributes for each stream or transaction specified. If -hidden is set, the visible attributes are in a list preceding the hidden attributes.

- -win <wave>

  (optional) If present, this option specifies the wave window for which the changes should apply. <wave> is the Tk name (not the title) for the wave window for which the changes apply.

- <stream> ...

  (required) The path to a transaction stream. Multiple streams are allowed as a space separated list. No wildcards are allowed. The tr order command requires either a stream or unique ID.

Examples

- Set the order in which attributes appear in the wave window for a specified transaction stream:

  tr order -attr {attr1 attr2 attr3} /path/tr03

- Resets the attribute order for the /*top/stream* stream to the default order:

  tr order -default /top/stream

- Displays a Tcl list of visible attributes, followed by the hidden attributes, of the transaction stream */top/stream2*:

  tr order -visible -hidden /top/stream2

Related Topics

- Recording and Viewing Transactions
- tr color
- tr uid

tr uid

This command returns a list of unique transaction IDs for a specified time span on specified streams, or for all times on specified streams. A transaction UID is the logical name of its dataset and its a 64-bit serial number created during simulation.

Usage: you can pass the returned UIDs to the tr color command to specify a particular transaction.

The returned UIDs represent transactions that are ACTIVE during the time span. If a transaction starts anywhere in the time span, at the start of the span or even at the end of the span, it is considered active. A transaction that ends at the start time is not active.

The optional arguments in this command apply either to a:

- listing of transactions occurring over a large range of time (using -end and -start)
- listing of transactions that are active at one specific time (using -time)

Unique abbreviations are accepted for all arguments.

Syntax

tr uid [-time <time> <stream> ...] | [-start <time> -end <time> <stream> ...] | [<stream> ...]

Arguments

- -end

  (required in conjunction with the -start option) Specifies the start of the span of time from which UIDs should be obtained.

- <time>

  (required) Specifies time, or time and delta (must be a positive integer).

- -start

  (required in conjunction with the -end option) Specifies the start of the span of time from which UIDs should be obtained.

- -time

  (required unless -end and -start are specified.) Specifies a point in time for the command.

- <stream> ...

  (required) The path to a transaction stream. If specified without any other arguments, a list of all UIDs on the specified stream are reported. May specify more than one stream. No wildcards are allowed.

Examples

- List all transaction UIDs for a specified transaction stream:

  tr uid /path/tr03

- List the transaction UID for a specified transaction stream at a particular time:

  tr uid -time 20ns {sim 209456}

Related Topics

- Recording and Viewing Transactions
- tr color
- tr order

transcribe

This command displays a command in the Transcript window, and then executes the command.

The transcribe command is normally used to direct commands to the Transcript window from an external event such as a menu pick or button selection. The add button and add_menuitem commands can use transcribe. The transcribe command can be used to force a command to be echoed.

Returns nothing.

Syntax

transcribe **<command>**

Arguments

- <command>

  (required) Specifies the command to execute.

Examples

- Create a button labeled "pwd" that invokes transcribe with the pwd Tcl command, and echoes the command and its results to the Transcript window. The button remains active during a run.

```
add button pwd {transcribe pwd} NoDisable
```

Related Topics

- add button
- add_menuitem

transcript

This command controls echoing of commands executed in a macro file.

If no option is specified, the current setting is reported.

Syntax

transcript [on | off | -q | quietly]

Arguments

- on

  (optional) Specifies that commands in a macro file will be echoed to the Transcript window as they are executed.

- off

  (optional) Specifies that commands in a macro file will not be echoed to the Transcript window as they are executed.

- -q

  (optional) Returns "0" if transcripting is turned off or "1" if transcripting is turned on. Useful in a Tcl conditional expression.

- quietly

  (optional) Turns off the transcript echo for all commands. To turn off echoing for individual commands see the quietly command.

Examples

- Commands within a macro file will be echoed to the Transcript window as they are executed.

  transcript on

- If issued immediately after the previous example, the message:

  transcript

  Returns

  ```
  Macro transcripting is turned ON.
  ```

Related Topics

- Transcript Window
- echo
- transcribe
- .main clear

transcript file

This command sets or queries the current PrefMain(file) Tcl preference variable. You can use this command to clear a transcript in batch mode or to limit the size of a transcript file. It offers an alternative to setting the PrefMain(file) Tcl preference variable through the GUI.

Syntax

transcript file [<filename>]

Arguments

- <filename>

  (optional) Specifies a name for the transcript file. Wildcard characters are allowed, and “stdout” or “stderr” are valid file names. If you specify a new file, the existing transcript file is closed and a new transcript file opened. If you specify an empty string (""), the existing file is closed and no new file is opened. If you don’t specify this argument, the current filename is returned.

Note

You can prevent overwriting older transcript files by including a pound sign (#) in <filename> when <filename> is a repeated string. The simulator replaces the pound character (#) with the next available sequence number when saving a new transcript file.

Examples

- Close the current transcript file and stops writing data to the file. This is a method for reducing the size of your transcript.

  ```
  transcript file ""
  ```

- Close the current transcript file named *trans1.txt and* open a new transcript file, incrementing the file name by 1.

  ```
  transcript file trans#.txt
  ```

  Closes *trans1.txt* and opens *trans2.txt.*

- This series of commands results in the transcript containing only data from the second millisecond of the simulation. The first transcript file command closes the transcript so no data is being written to it. The second transcript file command opens a new transcript and records data from 1 ms to 2 ms.

  ```
  transcript file ""
  run 1 ms
  transcript file transcript
  run 1 ms
  ```

Related Topics

- "Creating a Transcript File"
- “Setting GUI Preferences”
- Transcript Window
- .main clear
- transcript path
- transcript sizelimit

transcript path

This command returns the full pathname to the current transcript file.

Syntax

transcript path

Arguments

None

Related Topics

- "Creating a Transcript File"
- “Setting GUI Preferences”
- “Transcript Window”
- transcript file

transcript sizelimit

This command sets or queries the current preference value for the transcript fileSizeLimit value. If the size limit is reached, the transcript file is saved and simulation continues.

Syntax

transcript sizelimit [<size>]

Arguments

- <size>

  (optional) Specifies the size, in KB, of the transcript file where the default is 0 or unlimited. The actual file size may be larger by as much as one buffer size (usually about 4k), depending on the operating system default buffer size and the size of the messages written to the transcript.

Note

You can set the size of the transcript file with the $PrefMain (fileSizeLimit) Tcl variable in the Preferences dialog. Refer to "Setting GUI Preferences" for more information.

Related Topics

- "Creating a Transcript File"
- "Setting GUI Preferences"
- "Transcript Window"
- transcript file

triage dbfile

This command can be run from either the shell command line or the vsim command prompt to create a database (*questasim.tdb*) of test data and/or simulation messages extracted from a:

- a UCDB
- a WLF file
- a LOG file

By default, the triage database file (TDB) is cumulative: tests and/or messages added by previous commands are preserved. The -clear switch can be used to remove existing data before adding the extracted data into the TDB.

Messages associated with tests which are already represented in the database are not inserted into the database unless -append or -force is specified.

The triage dbfile command imports user-defined attributes from the test data record(s) in a UCDB file into the TDB file. These attributes can be displayed as columns in the Results Analysis window and used in hierarchy configurations and filter expressions. They are not supported in the text report (ie: the triage report command). To support the user-defined attribute table, the triage query command supports a "-attributes" option and direct SQL queries passed to that command may use the "(%ATTRIBUTES%)" token to refer to the attributes table. In addition, when a UCDB is imported by the triage dbfile command, the number of user-defined attributes added to the TDB file will be reported in the status message.

When tests are extracted from a UCDB file, the triage dbfile command attempts to find the WLF (or plain-text LOG) file associated with each test. The WLFNAME test data record attribute (or the attribute specified by -wlfattr) is used to locate the relevant WLF file. If no WLF file is specified (or if the WLF file cannot be found), the LOGNAME test data record attribute (or the attribute specified by -logattr) is used to locate a plain-text LOG file containing messages from the simulation. Any specified LOG file is disregarded as the source for import if a WLF file is found.

If the WLF/LOG file associated with one of the imported tests is found and imported, and that same file is also listed on the command line (or in the "inputs file"), a warning is issued and the redundant file name is ignored.

Each imported message can be associated with a "testname". If you are not using a UCDB file, you must use the -testname argument in order to associating a testname with the data. If -testname is specified, you can only specify for import one of either a WLF file or a plain-text LOG file for import (not both). If you don’t use -testname, any number of WLF and/or LOG files may be imported, in which case the basename of each file (i.e.: the file name without the ".wlf" or other extension) becomes the testname under which messages from that file are stored. Tests imported from WLF and/or plain-text LOG files imported as part of a UCDB test data import will automatically be associated with the testname of the UCDB test data record that points to that file.

A “triage dbfile -rulesfile <rulesfile>” command “transforms” messages for ease of viewing and querying purposes. The -rulesfile option is required for importing plain-text LOG files. See “Message Transformation” for further information.

Syntax

triage dbfile <inputoptions> {<testoptions> | (<msgoptions1> | <msgoptions2)}
[-help] [-verbose] [-debug] [-clear] [-name <tdb_filename>] [-deletemsg <globstr>]

where:

<inputoptions>

Arguments that apply to the processing of all data being transformed:

[-append] [-buflines <int>] [-force] [-ignorewlf] [-inputsfile <file>] [-nowrite]
[-rulesfile <rulesfilename> [-rulesfile <rulesfilename2>]...]

<testoptions>

Arguments which apply to the processing of the input UCDB files:

[<UCDB_file>] [-teststatus {F|E|W|P|M|S}] [-teststatusAll]
[[-wlfattr <str>] [-logattr <str>]] [<ucdbfilename> [<ucdbfilename>]...]

<msgoptions1>

Arguments when specifying a testname

-testname <testname> {<wlffilename> | <logfilename>}
[-severity {I|F|E|W|N}] [-severityAll]

<msgoptions2>

Arguments when not specifying a testname:

[<wlffilename> [<wlffilename>]...] [<logfilename> [<logfilename>]...]
[-severity {I|F|E|W|N}] [-severityAll]

Arguments

- -append

  (optional) Appends new data to existing data when data associated with a specified testname is already represented in the database.

- -buflines <int>

  (optional) Default is 10. This determines how many input LOG (*.log*) file lines are in the buffer at any given time; relevant when using a transformation rulesfile. Controls the maximum number of lines a single message may occupy and still be detected.

- -clear

  (optional) Clears data from an existing database. The default behavior, where -clear is not invoked, is that "triage dbfile" commands insert new data into the database.

- -debug

  (optional) Prints every message extracted from each WLF or plain-text LOG file during transformation (also requires -rulesfile). This option has no effect on UCDB files.

- -deletemsg <globstr>

  (optional) Deletes messages from the database where the specified <globstr> is found in the message string.

- -force

  (optional) When importing data associated with a testname that is already represented in the database, this option causes the deletion of the testname and any data associated with that testname from the database prior to importing the new data.

- -ignorewlf

  Forces the triage dbfile command to import simulation messages from the LOG file instead of from the WLF file. The default behavior is for triage dbfile to give preference to the WLF file if valid message data is found, and only use the LOG file if valid message data is missing from the WLF file. Only effective when importing a UCDB file.

- -inputsfile <file>

  (optional) Specifies the path to a file containing a list of *.ucdb*, *.wlf*, and/or plain text LOG files. The <file> specified should contain one input file per line, each of which can include path variables, such as in the following: *$MYPATH/test1.ucdb*.

- -logattr <str>

  (optional) Applies only to UCDB files. The test data record attribute in the specified UCDB whose value is the corresponding log file name. The default keyword is LOGNAME. Once the "-logattr" is set, that value applies to every UCDB on the same command line.

- <logfilename> [<logfilename>]...

  (optional) Specifies the name and/or path to a plain-test LOG file.

- -name <tdb_filename>

  (optional) Names the triage database file. Default filename is *questasim.tdb*. When the command is run from the vsim prompt, the tdb_filename specified with -name becomes “sticky” in that future commands applied will use the filename specified with the initial -name argument until it is explicitly set to another name. Howeveer, a previous -name setting is not preserved for commands issued at the command shell level.

- -nowrite

  (optional) Suppresses the insertion of the tests and/or messages into the database. Transformation processing still occurs. Used in conjunction with -verbose or -debug to debug transform constructs. States the name of the matching transform and the file:line in the rules file where the transform was found.

- -rulesfile <rulesfilename> [-rulesfile <rulesfilename2>]...

  (required only for importing plain-text LOG files; optional otherwise) Specifies a transformation rules file (applies only to *.wlf* or *.log* files). If the -rulesfile argument is included on the command line, the specified file is parsed for message transformation constructs. If at least one transform construct is found, then all messages destined for insertion into the TDB file are compared against the transforms found in the transformation rules file. TCL commands and variables are allowed in rules files.

  Multiple rules files are allowed: Each message is checked against the transform rules in each rules file in the order they appear on the command line.

  Messages imported from a WLF file which do not match any transformation expressions are inserted into the TDB file as-is, with the severity specified in the WLF file. Messages imported from a plain-text LOG file which do not match any transformation expressions are dropped. Messages whose transformation fails to assign a valid severity to the message are dropped as well.

- -severity {I|F|E|W|N}

  (optional) Default behavior is “-severity IFE” whereby only internal error, fatal and error messages are saved. Only WLF messages that match this specified severity are inserted into the triage database.

  I -- internal error

  F -- fatal, failure

  E -- error

  W -- warning

  N -- note

- -severityAll

  (optional) Basically, the same as -severity IFEWN.

- -testname <testname> {<wlffilename> | <logfilename>}

  (optional) This testname can be used to override other methods of determining the testname for message(s). When -testname is specified, only one WLF and/or plain-text LOG input files is allowed. If the -testname argument is not provided, the testname is determined from the UCDB, or extracted from the input filename. For example, if the input filename is *random22.wlf*, then “random22” would be used as the testname.

- -teststatus {F|E|W|P|M|S}

  (optional) Specifies one or more test status codes used to control the automatic import of a WLF or LOG file associated with each test. Default behavior is -teststatus FE. If WLF files are read using "wlfattr", such as test attribute WLFNAME, then the test status is checked first before proceeding. Using the default settings, this means that only tests that had a fatal or error status would have corresponding WLF messages loaded into the database.

  F -- fatal

E -- error

W -- warning

P -- pass or OK

M -- merge error

S -- missing

- -teststatusAll

  (optional) Basically, the same as -teststatus FEWP.

- <ucdbfilename> [<ucdbfilename>]...

  (required if neither a WLF nor a LOG file is specified) Specifies the name and/or path to one or more Unified Coverage DataBase (UCDB) filename(s).

- -wlfattr <str>

  (optional) This argument applies only to UCDB files. The test data record attribute in the UCDB whose value is the location of the WLF file associated with each test. The default test attribute name is WLFNAME. Once the "-wlfattr" is set, the value applies to every UCDB on the same command line.

- <wlffilename> [<wlffilename>]...

  (required if neither UCDB nor LOG file is specified; optional otherwise) Specifies the name and/or path of one or more WLF file names.

- -verbose

  (optional) Prints additional information to the log output.

Examples

- Let's say you stored the path to the WLF file under the attribute WLFFILENAME instead of WLFNAME in the *example1.ucdb* file. Enter the following command for automatic WLF import:

  triage dbfile -clear -wlfattr WLFFILENAME example1.ucdb

  The *example1.ucdb* data is imported first. Then, the command looks for UCDB attribute name WLFFILENAME whose value is *example1.wlf*. and it imports that data.

Related Topics

- "Analyzing Verification Results"
- triage dbinsert
- triage report
- triage query
- triage report
- triage view

triage dbinsert

This command, which can be run from either the shell command line or from the vsim prompt, places one or more messages directly into a triage database (TDB) file. When inserted, the messages are handled in the same way as a single message from a WLF or plain-text LOG file:

1. Each message is transformed if a transform rules file is specified with the -rulesfile option.
2. The severity is checked against the specified severity filter (-severity).
3. The message is inserted into the TDB file unless -nowrite is specified.

The “triage dbinsert” command assumes the messages are being appended to any other messages that may be associated with the same testname and, therefore, does not check for existing messages associated with the same testname.

Syntax

triage dbinsert [-verbose] [-debug] [-name <tdb_filename>] [-nowrite]
[-rulesfile <rulesfilename> [-rulesfile <rulesfilename2>]]
[-setseverity <string>] [-settime <int_time>] [-severity {I|F|E|W|N}] [-severityAll]
[-testname <testname>] <message> [<message>]...

Arguments

- -debug

  Optional. Prints every message extracted from each WLF or plain-text LOG file during transformation (also requires -rulesfile). This option has no effect on UCDB files.

- <message> [<message>]...

  Required. The message(s) to be inserted. If the message string contains spaces or other characters that could be misinterpreted by the command, surround the string by quotes.

- -name <tdb_filename>

  Optional. Specifies an path and file name for the *.tdb* file. Default filename is *questasim.tdb*.

- -nowrite

  Optional. Suppresses the insertion of the message(s) into the database. Transformation processing still occurs. This switch is used to debug transform regular expressions. When enabled, the same level of detailed information is output as is shown when the -verbose switch is used. When used in combination with -verbose, an even greater level of detail is output. See examples.

- -rulesfile <rulesfilename> [-rulesfile <rulesfilename2>]

  Optional. Specifies the path to the transformation rules file. Multiple rules files are allowed: Each message is checked against the transform rules in each rules file in the order they appear on the command line.

- -setseverity <string>

  Optional. Sets the severity code used for the inserted messages. The default value is "Normal".

- -settime <int_time>

  Optional: Sets the simulation time of the inserted messages. The time should be an integer followed immediately by one of the recognized time unit identifiers. A single optional underscore is allowed between the numeric value and the time unit for readability. The default value is "0ns".

- -severity {I|F|E|W|N}

  Optional. Specifies one or more message severity codes used to filter messages from inclusion in the database. A table of message severity codes is given in the Message Severity section above. If the severity code of any given message does not match one of the codes specified by this option, that message will not be imported into the database. Default behavior is -severity IFEWN (that is, messages with all known message severity codes are accepted).

- -severityAll

  (optional) Displays messages with all known message severity codes. Identical behavior to -severity IFEWN.

- -testname <testname>

  (optional) Specifies the testname of the data to be inserted.

- -verbose

  (optional) Prints additional information to the screen.

Related Topics

- "Analyzing Verification Results"
- triage dbfile
- triage report
- triage query
- triage report
- triage view

triage dbrefresh

This command clears the old triage database, and re-creates the database using a saved history of previous "triage dbfile" and “triage dbinsert” commands. It can be run from either the command shell or the vsim command prompt.

Syntax

triage dbrefresh [-name <tdb_filename>] [-verbose]

Arguments

- -name <tdb_filename>

  Optional. Specifies a path and file name for the *.tdb* file to be refreshed. Default filename is *questasim.tdb* located in the current directory.

- -verbose

  Optional. Prints additional information to the screen.

Related Topics

- “Analyzing Verification Results”
- triage dbfile
- triage dbinsert
- triage query
- triage view
- triage report

triage query

Provides programmatic access to the contents of the triage database (TDB) for debug and scripting purposes. This command can be run from either the shell command line or the vsim command prompt. The command has two forms:

- You can use -statement to specify one or more SQL query statement to be run against the triage database. The results are returned as a TCL "list-of-lists", which is a list of records where each list member is a list of field results. You can specify certain tables within the TCL code using aliases. See Table 2-10 for a list of alias names.
- If no query is specified with -statement, the triage query command returns the contents of any specified tables of data from the TDB. If no tables are specified, the contents of all tables is returned. Each table's contents are in the same "list-of-lists" format, and all fields in each table are included in the returned result. If data from multiple tables are returned, the list table data is shown in the order listed in Table 2-10.

Table 2-10. Alias for Tables

Table data returned...	Alias (for use with -statement)	Argument (use without -statement)
Attribute (user-defined)	%ATTRIBUTE%	-attributes
Field (user-defined)	%FIELDS%	-fields
History	%HISTORY%	-history
Message	%MESSAGES%	-messages
Properties	%PROPERTIES%	-properties
Test	%TESTS%	-tests

Syntax

triage query [-help] [-name <tdb_filename>] <options>

where:

<options> = -analyze | <sql_options> | <query_options>

<sql_options> = -statement <query_string> [-elapsed]

<query_options> = [-files] [[-unique | -verbose] <table_options>]

<table_options> = [-attributes] [-fields] [-history] [-messages] [-tests] [-properties]

Arguments

- -analyze

  Optional: Returns a number of record counts to help estimate the complexity of the TDB file contents.

- -attributes

  (optional) Returns a list-of-lists dump of user-defined attributes.

- -history

  (optional) Returns a list-of-lists dump of previously executed triage dbfile and/or triage dbinsert commands (i.e. the "history" table).

- -elapsed

  (optional) Returns the amount of time a given query took (for investigating database performance).

- -fields

  (optional) Returns a list-of-lists dump of the contents of the user-defined field table.

- -files

  Optional. Returns a list containing: a list of WLF/LOG files represented in the database, and a list of UCDB files represented in the database.

- -messages

  Optional. Returns a list-of-lists dump of the contents of the message table.

- -name <tdb_filename>

  Optional. Specifies the path/filename of the triage database (TDB) file. The default filename is *default.tdb*.

- -properties

  Optional. Returns the database properties (i.e.: the contents of the "properties" table).

- -statement <query_string>

  Optional. Specifies an SQL query string to be applied to the database. This argument may appear multiple times in a single command. If multiple -statement arguments are specified, the triage query command returns a list of result tables in the order in which the query statements appeared on the command line. Mutually exclusive with all other command arguments besides -elapsed.

- -tests

  Optional. Returns a list-of-lists dump of the contents of the tests table (i.e.: a list of imported UCDB test data records).

- -unique

  Optional. Returns a list of unique values for the main field in that table (i.e. msg in the case of the message table, testname in the case of the test table, and so on).

- -verbose

  Optional. In conjunction with any of the dumping arguments, it returns the table data in a more human-readable format.

Examples

- Query the TDB (default is *./questasim.tdb*) with a specific SQL statement, and return a list of messages from the message table with a severity of “Error”.

  triage query -statement “SELECT mgs, testname FROM %MESSAGES% WHERE severity == 'Error'”

- Query the TDB and return a list of data from the Tests table.

  triage query -tests

Related Topics

- “Analyzing Verification Results”
- triage dbfile
- triage dbinsert
- triage dbrefresh
- triage report
- triage view

triage report

This command reads in a triage database (*.tdb*) and prints out a text report, after applying specified filters. It can be run from either the shell command line or the vsim command prompt.

Messages in the triage report are listed from highest level of severity to lowest: messages with fatal or error severity are shown first such that the user can more easily focus on highest priority failures.

Syntax

triage report [-verbose] [-name <tdb_filename>] [-append] [-file <outputfilename>] <filteroptions> <formatoptions>

where:

<filteroptions> =

[-teststatus [F|E|W|P|M|S]] [-teststatusAll] [-severity [I|F|E|W|N]] [-severityAll] [-message <globstr>] [-excludemsg <globstr>]

<formatoptions>

[-nummsg <int>] [-concise [T|F] [-notruncation] [-details [M|A|T|U]] [-detailsAll]

Arguments

- -append

  Optional. Appends to output filename, or else the old output filename is overwritten.

- -concise [T|F]

  Optional. Displays additional information to the concise rows-and-columns format or the line-by-line -notruncation format. This argument is ignored if -details argument is used.

  T - testname

  F - filename

- -details [M|A|T|U]

  Optional. Displays the following detailed information.

  M -- message

  A -- assertion

  T -- test record

  U -- user-defined

- -detailsAll

  Optional. Same as -details MATU.

- -excludemsg <globstr>

  Optional. Filter the display - do not show messages where <globstr> is found in the message string.

- -file <outputfilename>

  Optional. Specify output filename, or else the report is sent to stdout.

- -message <globstr>

  Optional. Filter the display - only shows messages where <globstr> is found within the message string.

- -name <tdb_filename>

  Optional. Default filename is *questasim.tdb*.

- -notruncation

  Optional. Do not truncate display of minimal default data. Output is converted to line-by-line instead of rows-and-columns. This argument is ignored if -details argument is used.

- -nummsg <int>

  Optional. Limit the maximum number of the same message to the number specified. Default behavior is “1”, where all identical messages are compressed into one message with the earliest time printed. The value of <int> must be a positive integer (“0” is ignored with a warning generated). When <int> is an integer greater than 1, the same message prints out the specified number of times, where each message displays the time it was issued.

- -severity [I|F|E|W|N]

  Optional. Filter the display — only show messages with the following severity. Default behavior is -severity IFEWN

  I -- internal error

  F -- fatal, failure

  E -- error

  W -- warning

  N -- note

- -severityAll

  Optional. Displays messages with all known message severity codes. Identical behavior to -severity IFEWN.

- -teststatus [F|E|W|P|M|S]

  Optional. Filter the display — show only messages where the tests have the specified status. Default behavior is -teststatus FEWPMS.

  F -- fatal

  E -- error

  W -- warning

  P -- pass or OK

  M -- merge error

S -- missing

If the filtering matches, then the test record information is printed. Test record information is displayed only when -details T is also specified.

Also, if a UCDB file specified as the input file does not pass the teststatus filter, then the WLF or Log file that would have been automatically imported is discarded instead. This affects only the automatic import of WLF or Log files via a UCDB file.

- -teststatusAll

  Optional. Specifies -teststatus FEWP.

- -verbose

  Optional. Prints additional information to the screen.

Related Topics

- "Analyzing Verification Results"
- triage dbfile
- triage dbinsert
- triage dbrefresh
- triage query
- triage view

triage view

This command loads a triage database (*.tdb*) into the Verification Results Analysis window, and must be invoked from the vsim (Transcript window) command prompt. This is in contrast to all other triage commands, which may be run from either the vsim command prompt or the shell prompt.

Multiple Results analysis windows can be opened at any time.

Syntax

triage view [-name <tdb_filename>] [<viewoptions>]

where:

<viewoptions>

[-collayout <name>] [-filterexp <name>] [-hierconfig <name>] [-question <name>] [-sortconfig <config_name>]

Arguments

- -collayout <name>

  Optional. Specifies the previously named and saved column layout to be used in the new window.

- -filterexp <name>

  Optional. Specifies the previously named and saved filtered expression to be used in the new window.

- -hierconfig <name>

  Optional. Specifies the previously named and saved hierarchy configuration to be used in the newly opened window.

- -name <tdb_filename>

  Optional. Specifies the name of the triage database to be opened with the command. If -name is not specified, the default database name of *questasim.tdb* is loaded into the Results Analysis window.

- -question <name>

  Optional. Select the named triage question in the new window.

- -sortconfig <config_name>

  Optional. Applies a pre-defined sort configuration to the newly opened window. Sorting configurations are those defined in the the Results Analysis window's Sort Configuration dialog or pulldown list.

Related Topics

- "Analyzing Verification Results"
- triage dbfile
- triage dbinsert
- triage dbrefresh
- triage query

tssi2mti

This command is used to convert a vector file in TSSI Format into a sequence of force and run commands.

The stimulus is written to the standard output.

The source code for tssi2mti is provided in the *examples* directory as:

```
<install_dir>/examples/tssi2mti/tssi2mti.c
```

Syntax

tssi2mti <signal_definition_file> [<sef_vector_file>]

Arguments

- <signal_definition_file>

  (required) Specifies the name of the TSSI signal definition file describing the format and content of the vectors.

- <sef_vector_file>

  (optional) Specifies the name of the file containing vectors to be converted. If none is specified, standard input is used.

Examples

- The command will produce a do file named *trigger.do* from the signal definition file *trigger.def* and the vector file *trigger.sef*.

  tssi2mti trigger.def trigger.sef > trigger.do

- This example is the same as the previous one, but uses the standard input instead.

  tssi2mti trigger.def < trigger.sef > trigger.do

Related Topics

- force
- run
- write tssi

typespec

This command queries class names and class relationships of SystemVerilog classes.

This command and the find insource command are available for debugging, should you encounter design elaboration errors.

Syntax

typespec <pattern> [-isa <value>] [-class <value> | -ancestry <value>] [-exact] [-regexp] [-indent <value>]]

Arguments

- <pattern>

  (optional) The pattern you are querying for.

- -isa <value>

  (optional) Returns classes derived from the given <value>.

- -class <value>

  (optional) Returns specialized classes of the given <value>, where <value> represents a parameterized class and the results are specific instances of the class with given parameter values.

- -ancestry <value>

  (optional) Returns a list of base classes for the given <value>.

- -exact

  (optional) Returns results that match the given <pattern> exactly.

- -regexp

  (optional) Specifies that all <pattern> arguments should be treated as regular expressions.

- -indent <value>

  (optional) Prefixes each level of the class inheritance hierarchy with <value>.

Related Topics

- find insource
- DISABLE_ELAB_DEBUG environment variable

ui_VVMode

This command specifies behavior when encountering UI registration calls used by verification packages, such as AVM or OVM. Returns the current setting when specifies without an argument.

Syntax

ui_VVMode [full | logclass | logobj | nolog | off]

Arguments

- full

  (optional) Enables the context registration of the UI registration call and automatically logs both the class type and the registered object to the WLF file.

- logclass

  (optional) Enables the context registration of the UI registration call and automatically logs the class type of the registered object to the WLF file.

- logobj

  (optional) Enables the context registration of the UI registration call and automatically logs the registered object to the WLF file

- nolog

  (optional) Enables the context registration of the UI registration call, but does not automatically log the registration to the WLF file. (default)

- off

  (optional) Disables context registration and automatic logging when encountering UI registration calls.

Description

UI registration calls, Verilog system tasks specific to this product, are typically included in verification packages such as AVM and OVM so that key information about the packages is available when debugging the simulation. The UI registration calls include:

- $ui_VVInstallInst() — Defines a region in the context tree, which will appear in the Structure window.
- $ui_VVInstallObj() — Adds an object to the defined parent, which will appear in the Objects window when the parent instance is selected in the Structure window.
- $ui_VVInstallPort() — Adds a port that is an object that connects to another component, which will appear in the Objects window when the parent instance is selected in the Structure window.
- $ui_VVSetFilter() — Specifies which class properties should not be shown in the GUI.

- $ui_VVSetAllow() — Specifies which class properties should be retained that were filtered out with $ui_VVSetFilter.

unsetenv

This command deletes an environment variable. The deletion is not permanent – it is valid only for the current Questa SIM session.

Syntax

unsetenv <varname>

Arguments

- <varname>

  (required) The name of the environment variable you wish to delete.

Related Topics

- setenv
- printenv

up

This command searches for object transitions or values in the specified List window.

It executes the search on objects currently selected in the window, starting at the time of the active cursor. The active cursor moves to the found location.

Use this command to move to consecutive transitions or to find the time at which an object takes on a particular value, or an expression of multiple objects evaluates to true. Refer to the down command for related functionality.

Prerequisites

The procedure for using up entails three steps:

1. Click on the desired waveform.
2. Click on the desired starting location.
3. Issue the up command. (The seetime command can initially position the cursor from the command line, if desired.)

Returns: <number_found> <new_time> <new_delta>

Syntax

up [-expr {<expression>}] [-falling] [-noglitch] [-rising] [-value <sig_value>] [-window <wname>] [<n>]

Arguments

- -expr {<expression>}

  (optional) Searches the waveform display for an expression. When the search evaluates to a boolean true, the active cursor moves to the found location. The expression may involve more than one signal, but is limited to signals that have been logged in the referenced Wave window. A signal may be specified either by its full path or by the shortcut label displayed in the Wave window.

  {<expression>} — Any valid expression. See GUI_expression_format for the format of the expression. The expression must be placed within curly braces.

- -falling

  (optional) Searches for a falling edge on the specified object if that object is a scalar. If it is not a scalar, the option will be ignored. Optional.

- -noglitch

  Specifies that delta-width glitches are to be ignored.

- -rising

  (optional) Searches for a rising edge on the specified object if that object is a scalar. If it is not a scalar, the option will be ignored.

- -value <sig_value>

  (optional) Specifies a value of the signal to match. Only one signal may be selected, but that signal may be an array.

  <sig_value> — Must be specified in the same radix that the selected waveform is displayed. Case is ignored, but otherwise the value must be an exact string match -- don't-care bits are not yet implemented.

- -window <wname>

  (optional) Specifies an instance of the Wave window that is not the default. Otherwise, the default Wave window is used. Use the view command to change the default window.

  <wname> — The name of a Wave window other than the current default window.

- <n>

  (optional) Specifies to find the nth match. If less than n are found, the number found is returned with a warning message, and the marker is positioned at the last match.

Examples

- Find the last time at which the selected vector transitions to FF23, ignoring glitches.up

  up -noglitch -value FF23

- Go to the previous transition on the selected object.

  up

The following examples illustrate search expressions that use a variety of signal attributes, paths, array constants, and time variables. Such expressions follow the GUI_expression_format.

- Search up for an expression that evaluates to a boolean 1 when signal *clk* just changed from low to high and signal *mystate* is the enumeration reading and signal */top/u3/addr* is equal to the specified 32-bit hex constant.

  up -expr {clk'rising && (mystate == reading) && (/top/u3/addr == 32'habcd1234)}

- Search up for an expression that evaluates to a boolean 1 when the upper 8 bits of the 32-bit signal */top/u3/addr* equals hex ac.

  up -expr {(/top/u3/addr and 32'hff000000) == 32'hac000000}

- Search up for an expression that evaluates to a boolean 1 when logfile time is between 23 and 54 microseconds, clock just changed from low to high, and signal *mode* is enumeration writing.

  up -expr {((NOW > 23 us) && (NOW < 54 us)) && clk'rising && (mode == writing)}

Related Topics

- GUI_expression_format
- down
- seetime
- view

uvm call

This command calls a SystemVerilog UVM function, either from the uvm_pkg or in a user's UVM test bench code.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all -classdebug

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm call <function_name> <arg1> [<arg2>] …[<argN>]

Arguments

- <function_name>

  A UVM function name, including a path if the function is not in the current context. The path can be specified in the following ways:

  Class Instance Identifier — The CIID format is @<class-type>@<n> where <class_type> is the name of the class and <n> is the nth instance of that class. For example, *@my_class@3* is the third instance of the class type *my_class*. Class instances are only available after the design has been elaborated with a run 0 or after running the design. Refer to The Class Instance Identifier for more information.

Note

A CIID is unique for a given simulation. Modifying a design, or running the same design with different parameters, randomization seeds, or other configurations that change the order of operations, may result in a class instance changing. For example, *@packet@134* in one simulation run may not be the same *@packet@134* in another simulation run if the design has changed.

UVM Hierarchical Path — The UVM Hierarchical Path is in specified in the following format: <string>.<string>.<string>....<function_name>. For example, *uvm_test_top.middle.bottom0.my_obj*. that was registered for the object when the class was allocated via the UVM *create()* method. This is the path returned from a UVM *get_full_name()* function call and is referred to as a "UVM path."

Simulation Path — The Simulation Path is the hierarchical path mapping for the UVM function within the sim:/uvm_root context and is specified in the following format:

sim:/uvm_root/<string>/.../<function_name>. For example, s*im:/uvm_root/uvm_test_top/middle/bottom0/my_obj*. This hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters will match the simulator's path delimiter settings, and some component names may be slightly changed to use Verilog escaped identifiers if the component names do not conform to Verilog naming rules.

The command will look first into the current context to find the specified function. If the function does not exist in the current context, then the argument string will be parsed to determine if it contains one of the hierarchy specifications described above. In all cases it will check first to see if the path is a contains a CIID string. If it does not, then by default it will scan the path as a UVM-style path to the function. The default behavior can be changed to be simulation-style path names with the uvm mapmode command. Finally, even when the path name mapping mode is in the default UVM-style mode, it still can accept simulation-style paths if they start with a *sim:/uvm_root* prefix.

- <arg1> [<arg2>] …[<argN>]

  (optional) All arguments required by the function are specified in a space separated list in declaration order. If a function has default arguments, the arguments may be omitted from the command line provided that the arguments occur at the end of the declaration list. Function input arguments can be constant values including integers, enumerated values, and strings. A string containing spaces or special characters must be enclosed in double quotes (" ") or braces ({ }) or Tcl will try to interpret the string. For example: "my string" or {my string}. Arguments can also be design objects. Class object references can be arguments, and must be either an object in the current context, a class instance id string, or a simulation-style hierarchical pathname. If a UVM-style pathname is desired, the uvm uvmpath subcommand may be used. If a function has type inout, out, or ref arguments, suitable user design objects must be passed in as arguments. Any passed in argument will first be tested to determine if it is an appropriate constant value. If it is not, then the argument will be tested to determine if it is a design object. Consequently, where there is ambiguity between a constant string and the name of a design object, the constant will be given precedence. If in this case the design object is desired, the full hierarchical path to the object can be supplied to differentiate it from the constant string.

Examples

- Call text.e1_b.get_inst_count:

  uvm call test.e1_b.get_inst_count

- Call text.e1_b.get_inst_count using the simulator path, with an embedded Tcl call to uvm uvmpath:

  uvm call [uvm uvmpath sim:/uvm_root/test/e1_b].get_inst_count

Related Topics

- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm configtracing

This command enables or disables printing of configuration database reads and writes to the transcript as they occur. UVM configuration database reads and writes typically occur during the build phase of UVM elaboration, so turning uvm configtracing on or off should be done before UVM elaboration happens, which is before a "run 0" is issued. If specified without arguments returns the current tracing mode.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all

Syntax

uvm configtracing {0 | 1}

Arguments

- 0

  (default) Disables config database and resource database tracing.

- 1

  Enables config database and resource database tracing.

In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm ct.

Related Topics

- uvm call
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm displayobjections

This command returns the current objections to the transcript. If no argument is supplied, display objections for *uvm_top*.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all.

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm displayobjections [<uvm_object_path>]

Arguments

- <uvm_object_path>

  (optional) A UVM object name, including a path if the object is not in the current context. The path can be specified in the following ways:

  Class Instance Identifier — The CIID format is @<class-type>@<n> where <class_type> is the name of the class and <n> is the nth instance of that class. For example, *@my_class@3* is the third instance of the class type *my_class*. Class instances are only available after the design has been elaborated with a run 0 or after running the design. Refer to The Class Instance Identifier for more information.

Note

A CIID is unique for a given simulation. Modifying a design, or running the same design with different parameters, randomization seeds, or other configurations that change the order of operations, may result in a class instance changing. For example, *@packet@134* in one simulation run may not be the same *@packet@134* in another simulation run if the design has changed.

UVM Hierarchical Path — The UVM Hierarchical Path is in specified in the following format: <string>.<string>.<string>....<class_type>. For example, *uvm_test_top.middle.bottom0.my_obj*. that was registered for the object when the class was allocated via the UVM *create()* method. This is the path returned from a UVM *get_full_name()* function call and is referred to as a "UVM path."

Simulation Path — The Simulation Path is the hierarchical path mapping for the UVM object within the sim:/uvm_root context and is specified in the following format:

sim:/uvm_root/<string>/.../<class_type>. For example, *sim:/uvm_root/uvm_test_top/middle/bottom0/my_obj*. This hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters will match the simulator's path delimiter settings, and some component names may be slightly changed to use Verilog escaped identifiers if the component names do not conform to Verilog naming rules.

The command will look first into the current context to find the specified object. If the object does not exist in the current context, then the argument string will be parsed to determine if it contains one of the hierarchy specifications described above. In all cases it will check first to see if the path is a CIID string. If does not, then by default it will scan the path as a UVM-style path name. The default behavior can be changed to be simulation-style path names with the uvm mapmode command. Finally, even when the path name mapping mode is in the default UVM-style mode, it still can accept simulation-style paths if they start with a *sim:/uvm_root* prefix.

In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm do.

Examples

- Display the objections for /uvm_root/uvm_test_top/top_0/middle_2

  uvm displayobjections sim:/uvm_root/uvm_test_top/top_0/middle_2

  Returns:

```
#     Domain: common
#  Component: uvm_test_top.top_0.middle_2
#      Phase: common.run
#  Objection: run
# The total objection count is 2
# ---------------------------------------------------------
# Source  Total
# Count   Count   Object
# ---------------------------------------------------------
# 0       2             middle_2
# 1       1               bottom_0
# 1       1               bottom_1
# ---------------------------------------------------------
#
#     Domain: uvm
#  Component: uvm_test_top.top_0.middle_2
#      Phase: common.run
#  Objection: run
# The total objection count is 2
# ---------------------------------------------------------
# Source  Total
# Count   Count   Object
# ---------------------------------------------------------
# 0       2             middle_2
# 1       1               bottom_0
# 1       1               bottom_1
# ---------------------------------------------------------
```

Related Topics

- uvm call
- uvm configtracing
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm findregisters

This command lists the set of HDL registers attached to the UVM register models in a design. This list is used to create a file that allows visibility into the register blocks after optimization.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all -classdebug.

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm findregisters [-acc] **<uvm_reg_block_path>**

Arguments

- <uvm_reg_block_path>

  (required) Specifies the path to a UVM register block. The path can be specified in the following ways:

  UVM Path — (default) The UVM Path is specified in the following format: <string>.<string>.<string>....<uvm_component>. For example, *uvm_test_top.middle.bottom0.my_obj*.

  Simulation Path — (if uvm mapmode is set to simulation-style) The Simulation Path is the hierarchical path mapping for the UVM object within the *sim:/uvm_root* context and is specified in the following format: *sim:/uvm_root/<string>/.../<uvm_component>*. For example, *sim:/uvm_root/uvm_test_top/middle/bottom0/my_obj*. This hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters will match the simulator's path delimiter settings, and some component names may be slightly changed to use Verilog escaped identifiers if the component names do not conform to Verilog naming rules.

- -acc

  (optional) Returns the registers for the specified block in acc format for use with vlog and vopt.

Note

You can save the output of this command to a file with the following command:

exec echo [uvm findregisters -acc <uvm_reg_block_path>] > regaccfile.f

Examples

- List all registers under the register block *dut_rm* in acc format.

  uvm findregisters -acc sim:/uvm_root/uvm_test_top/dut_rm

  Returns:

  ```
  +acc=rn+/testbench/i_dut/apply_holding_reg
  +acc=rn+/testbench/i_dut/back_reg +acc=rn+/testbench/i_dut/cfg1
  +acc=rn+/testbench/i_dut/cfg2 +acc=rn+/testbench/i_dut/data_reg
  +acc=rn+/testbench/i_dut/front_reg +acc=rn+/testbench/i_dut/go_bit
  +acc=rn+/testbench/i_dut/holding_reg
  +acc=rn+/testbench/i_dut/my_vector +acc=rn+/testbench/i_dut/regA
  +acc=rn+/testbench/i_dut/regB +acc=rn+/testbench/i_dut/status_reg
  +acc=rn+/testbench/i_dut/thirtytwobit
  ```

- Save the list of registers under the register block *dut_rm* in acc format to the file *uvm_accfile.f*.

  ```
  exec echo {uvm findregisters -acc /
  sim:/uvm_root/uvm_test_top/dut_rm} > uvm_accfile.f
  ```

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm findsequences

This command returns the active sequences under a specified sequencer. Returns all active sequences when specified without arguments.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all -classdebug.

Elaborate the UVM testbench by executing the following command:

```
run 0
```

Syntax

uvm findsequences [<component_path>]

Arguments

- <component_path>

  (optional) Specifies the path to a UVM sequence. The asterisk wildcard (*) may be used to match all components. The path can be specified in the following ways:

  UVM Path — (default) The UVM Path is specified in the following format: <string>.<string>.<string>....<uvm_component>. For example, *uvm_test_top.middle.bottom0.my_obj*.

  Simulation Path — (if uvm mapmode is set to simulation-style) The Simulation Path is the hierarchical path mapping for the UVM object within the *sim:/uvm_root* context and is specified in the following format: *sim:/uvm_root/<string>/.../<uvm_component>*. For example, *sim:/uvm_root/uvm_test_top/middle/bottom0/my_obj*. This hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters will match the simulator's path delimiter settings, and some component names may be slightly changed to use Verilog escaped identifiers if the component names do not conform to Verilog naming rules.

Examples

- List all active sequences for the sequencer *uvm_test_top.e1_a.sequencer*.

  ```
  uvm findsequence uvm_test_top.e1_a.sequencer
  ```

  Returns a hypertexed list of the active sequences under the specified sequencer (Figure 2-5).

Figure 2-5. UVM Sequences

```
Transcript
CDBG 5> uvm findsequences uvm_test_top.e1_a.sequencer
# Sequencer: uvm_test_top.e1_a.sequencer
#    envA.sv:150:  uvm_test_top.e1_a.sequencer.seq  handle = @sequence_A_greatgrandpa
#    envA.sv:120:  uvm_test_top.e1_a.sequencer.ggp_seq  handle = @sequence_A_grandpar
#    envA.sv:82:  uvm_test_top.e1_a.sequencer.gp_seq  handle = @sequence_A_parent@2
#    envA.sv:39:  uvm_test_top.e1_a.sequencer.p_seq  handle = @sequence_A@2

CDBG 6>
```

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm handle

This command returns the simulation handle (class instance identifier or CIID) of the specified UVM object to the transcript, the same information returned by the examine -handle command. You must specify the -classdebug option with vsim to enable use of handles in other operations.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all -classdebug.

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm handle **<uvm_object_path>**

Arguments

- <uvm_object_path>

  (required) A UVM object name, including a path if the object is not in the current context. The path can be specified in the following ways:

  UVM Hierarchical Path — The UVM Hierarchical Path is in specified in the following format: <string>.<string>.<string>....<class_type>. For example, *uvm_test_top.middle.bottom0.my_obj*. that was registered for the object when the class was allocated via the UVM *create()* method. This is the path returned from a UVM *get_full_name()* function call and is referred to as a "UVM path."

  Simulation Path — The Simulation Path is the hierarchical path mapping for the UVM object within the sim:/uvm_root context and is specified in the following format: sim:/uvm_root/<string>/.../<class_type>. For example, s*im:/uvm_root/uvm_test_top/middle/bottom0/my_obj*. This hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters will match the simulator's path delimiter settings, and some component names may be slightly changed to use Verilog escaped identifiers if the component names do not conform to Verilog naming rules.

  The command will look first into the current context to find the specified object. If the object does not exist in the current context, then the argument string will be parsed to determine if it contains one of the hierarchy specifications described above. By default, it will scan the path as a UVM-style path name. The default behavior can be changed to be simulation-style path names with the uvm mapmode command. Finally, even when the path

name mapping mode is in the default UVM-style mode, it still can accept simulation-style paths if they start with a *sim:/uvm_root* prefix.

Examples

- Get the simulation handle for the UVM name path uvm_test_top.top_1.middle_2.bottom_0.env_h_1.agent_h.sequencer_h.

  uvm handle uvm_test_top.top_1.middle_2.bottom_0.env_h_1.agent_h.sequencer_h

  Returns:

  ```
  # @sequencer@4
  ```

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm mapmode

This command sets the UVM path mapping mode for UVM commands. You can set the simulator to accept either UVM "get_full_name" style paths or simulation "/uvm_root" style paths. If no argument is given, displays the current mapping setting.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all.

Elaborate the UVM testbench by executing the following command:

```
run 0
```

Syntax

uvm mapmode [0 | 1]

Arguments

- 0

  Simulator "/uvm_root" style paths are used. For example, *sim:/uvm_root/uvm_test_top/middle/bottom0/my_obj*.

- 1

  (default) UVM-style paths are used. For example, *uvm_test_top.middle.bottom0.my_obj*.

In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm mm.

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm printconfig

This command returns configuration information matching a specified component to the transcript. It also returns access information for a specified configuration object variable. Returns the global configuration table to the transcript if no argument is specified.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all.

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm printconfig [<uvm_component_path> [<config_obj_name>]]

Arguments

- <uvm_component_path>

  (optional) Specifies the path to a UVM object. The asterisk wildcard (*) may be used to match all components. The path can be specified in the following ways:

  UVM Path — (default) The UVM Path is specified in the following format: <string>.<string>.<string>....<uvm_component>. For example, *uvm_test_top.middle.bottom0.my_obj*.

  Simulation Path — (if uvm mapmode is set to simulation-style) The Simulation Path is the hierarchical path mapping for the UVM object within the *sim:/uvm_root* context and is specified in the following format: *sim:/uvm_root/<string>/.../<uvm_component>*. For example, *sim:/uvm_root/uvm_test_top/middle/bottom0/my_obj*. This hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters will match the simulator's path delimiter settings, and some component names may be slightly changed to use Verilog escaped identifiers if the component names do not conform to Verilog naming rules.

- <config_obj_name>

  (optional) Specifies a UVM configuration object variable name. Returns detailed information about access to that variable with respect to the given component.

In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm pc.

Examples

- Get configuration information for

  uvm printconfig uvm_test_top.top_1.middle_0.bottom_1.env_h_2.coverage_collector_h

  Returns:

```
# UVM_INFO @ 0:
uvm_test_top.top_1.middle_0.bottom_1.env_h_2.coverage_collector_h
[CFGPRT] visible resources:
# uvm_root::string_9 [] : (string) string_9
# -
# uvm_root::string_8 [] : (string) string_8
# -
# uvm_root::string_7 [] : (string) string_7
# -
# uvm_root::string_6 [] : (string) string_6
# -
# uvm_root::string_5 [] : (string) string_5
# -
# uvm_root::string_4 [] : (string) string_4
# -
# uvm_root::string_3 [] : (string) string_3
# -
# uvm_root::string_2 [] : (string) string_2
# -
# uvm_root::string_1 [] : (string) string_1
# -
# uvm_root::string_0 [] : (string) string_0
```

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm printfactory

This command returns UVM global factory information to the transcript. This command does not take any arguments

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all.

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm printfactory

In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm pf.

Arguments

None

Examples

- Print the current factory

  uvm uvmfactory

  Returns:

```
#
#### Factory Configuration (*)
#
#   No instance or type overrides are registered with this factory
#
# All types registered with the factory: 53 total
# (types without type names will not be printed)
#
#   Type Name
#   ---------
#   agent
#   bottom
#   config_object
#   coverage_collector
#   driver
#   env
```

```
#   middle
#   monitor
#   my_sequence
#   my_test
#   questa_uvm_recorder
#   sequencer
#   series_of_short_words_seq
#   short_words_seq
#   top_c
#   transaction
# (*) Types with no associated type name will be printed as
<unknown>
#
####
#
```

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm printstreams

This command returns a list of streams (transaction objects) to the transcript.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all.

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm printstreams [<uvm_component_path>]...

Arguments

- <uvm_component_path>

  (optional) Specifies a UVM object. The asterisk wildcard (*) may be used to match all components. The path can be specified in the following ways:

  UVM Path — (default) The UVM Path is specified in the following format: <string>.<string>.<string>....<uvm_component>. For example, *uvm_test_top.middle.bottom0.my_obj*.

  Simulation Path — (if uvm mapmode is set to simulation-style) The Simulation Path is the hierarchical path mapping for the UVM object within the *sim:/uvm_root* context and is specified in the following format: sim:/uvm_root/<string>/.../<uvm_component>. For example, s*im:/uvm_root/uvm_test_top/middle/bottom0/my_obj*. This hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters will match the simulator's path delimiter settings, and some component names may be slightly changed to use Verilog escaped identifiers if the component names do not conform to Verilog naming rules.

  In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm ps.

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm printtopology

This command returns the UVM global testbench topology to the transcript. This command does not take any arguments.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all.

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm printtopology

In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm pt.

Arguments

None

Examples

- Return the UVM testbench topology to the transcript.

  uvm printtopology

  Returns:

```
# UVM_INFO @ 0: reporter [UVMTOP] UVM testbench topology:
# ------------------------------------------------------------
# Name                        Type              Size  Value
# ------------------------------------------------------------
# uvm_test_top               my_test             -    @577
#   top_0                     top_c              -    @584
#    middle_0                 middle             -    @642
#      bottom_0               bottom             -    @707
#        env_h_0              env                -    @766
#          agent_h            agent              -    @832
#            driver_h         driver             -    @901
#              rsp_port        uvm_analysis_port  -   @916
#              recording_detail integral          32   'd1
```

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm setverbosity
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm setverbosity

This command sets the global reporting verbosity.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all.

Syntax

uvm setverbosity <uvm_report_verbosity_level> [<uvm_component_path>]

Arguments

- <uvm_report_verbosity_level>

  (required) UVM verbosity enum values, decimal integers, and optional simple expressions containing +'s and -'s, for example:

  UVM_LOW, UVM_HIGH+1, 99.

- <uvm_component_path>

  (optional) Specifies a UVM object. The asterisk wildcard (*) may be used to match all components. The path can be specified in the following ways:

  UVM Path — (default) The UVM Path is specified in the following format: <string>.<string>.<string>....<uvm_component>. For example, *uvm_test_top.middle.bottom0.my_obj*.

  Simulation Path — (if uvm mapmode is set to simulation-style) The Simulation Path is the hierarchical path mapping for the UVM object within the *sim:/uvm_root* context and is specified in the following format: sim:/uvm_root/<string>/.../<uvm_component>. For example, s*im:/uvm_root/uvm_test_top/middle/bottom0/my_obj*. This hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters will match the simulator's path delimiter settings, and some component names may be slightly changed to use Verilog escaped identifiers if the component names do not conform to Verilog naming rules.

  In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm sv.

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm simpath
- uvm traceobjections
- uvm uvmpath

uvm simpath

This command maps the path of a UVM hierarchical component to a simulator "/uvm_root" context hierarchy string.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all.

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm simpath <uvm_component_path>

Arguments

- <uvm_component_path>

  (required) Specifies a UVM object path name.

  UVM Path — The UVM Path is specified in the following format: <string>.<string>.<string>....<uvm_component>. For example, *uvm_test_top.middle.bottom0.my_obj*.

  In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm sp.

Examples

- Find the simulator path for the UVM component: uvm_test_top.top_0.middle_0.bottom_0.env_h_2.agent_h.sequencer_h.seq_2_1

  uvm simpath uvm_test_top.top_0.middle_0.bottom_0.env_h_2.agent_h.sequencer_h.seq_2_1

  Returns

  ```
  #
  sim:/uvm_root/uvm_test_top/top_0/middle_0/bottom_0/env_h_2/agent_h/
  sequencer_h/seq_2_1
  ```

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm traceobjections
- uvm uvmpath

uvm traceobjections

This command enables or disables objections tracing for a specified UVM component instance.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. For help with a particular subcommand, enter uvm help <subcommand>. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all.

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm traceobjections <uvm_component_path> [0 | 1]

Arguments

- <uvm_component_path>

  (required) Specifies a UVM object. The asterisk wildcard (*) may be used to match all components. If specified without the second argument, returns the current trace objection setting for the specified component to the transcript.

  The path can be specified in the following ways:

  UVM Path — (default) The UVM Path is specified in the following format: <string>.<string>.<string>....<uvm_component>. For example, *uvm_test_top.middle.bottom0.my_obj*.

  Simulation Path — (if uvm mapmode is set to simulation-style) The Simulation Path is the hierarchical path mapping for the UVM object within the *sim:/uvm_root* context and is specified in the following format: sim:/uvm_root/<string>/.../<uvm_component>. For example, s*im:/uvm_root/uvm_test_top/middle/bottom0/my_obj*. This hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters will match the simulator's path delimiter settings, and some component names may be slightly changed to use Verilog escaped identifiers if the component names do not conform to Verilog naming rules.

- 0 | 1

  (optional) Toggles objections tracing for the UVM component instance specified with <uvm_component_path>.

  0 — (default) Disable objections tracing

  1 — Enable objections tracing

In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm to.

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm uvmpath

uvm uvmpath

This command maps the path of a UVM component represented in the simulator "/uvm_root" hierarchy or in handle form back into a UVM full path name, similar to that returned from a UVM "get_full_name" function.

Note

This command is one of a group of Tcl commands useful for debugging UVM-based designs. You can find additional information about the group of commands by entering uvm on the command line. Refer to Related Topics below for links to the other subcommands.

Prerequisites

Simulate with vsim -uvmcontrol=all.

Elaborate the UVM testbench by executing the following command:

run 0

Syntax

uvm uvmpath <questa_uvm_component_path>

Arguments

- <questa_uvm_component_path>

  (required) A hierarchical path to an initialized UVM SystemVerilog class object. The path can be specified in the following ways:

  Simulation Path — The Simulation Path is the hierarchical path mapping for the UVM object within the sim:/uvm_root context and is specified in the following format: sim:/uvm_root/<string>/.../<class_type>. For example, *sim:/uvm_root/uvm_test_top/middle/bottom0/my_obj*. This hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters will match the simulator's path delimiter settings, and some component names may be slightly changed to use Verilog escaped identifiers if the component names do not conform to Verilog naming rules.

In addition to the full command syntax, you may abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to Command Shortcuts for more information. You can also use the command shortcut uvm up.

Examples

uvm uvmpath sim:/uvm_root/uvm_test_top/top_0/middle_0/bottom_0/env_h_2/agent_h/sequencer_h/seq_2_1

Returns:

```
# uvm_test_top.top_0.middle_0.bottom_0.env_h_2.agent_h.sequencer_h.
seq_2_1
```

Related Topics

- uvm call
- uvm configtracing
- uvm displayobjections
- uvm findregisters
- uvm findsequences
- uvm handle
- uvm mapmode
- uvm printconfig
- uvm printfactory
- uvm printstreams
- uvm printtopology
- uvm setverbosity
- uvm simpath
- uvm traceobjections

vcd add

This command adds the specified objects to a VCD file.

The allowed objects are Verilog nets and variables and VHDL signals of type bit, bit_vector, std_logic, and std_logic_vector (other types are silently ignored). The command works with mixed HDL designs.

All vcd add commands must be executed at the same simulation time. The specified objects are added to the VCD header and their subsequent value changes are recorded in the specified VCD file. By default all port driver changes and internal variable changes are captured in the file. You can filter the output using arguments detailed below.

Related Verilog tasks: $dumpvars, $fdumpvars

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

vcd add [-dumpports] [-file <filename>] [[-in] [-out] [-inout] | [-ports]] [-internal] [-r | -r -optcells] **<object_name> ...**

Arguments

- -dumpports

  (optional) Specifies port driver changes to be added to an extended VCD file. When the **vcd dumpports** command cannot specify all port driver changes that will appear within the VCD file, multiple vcd add -dumpports commands can be used to specify additional port driver changes.

- -file <filename>

  (optional) Specifies the name of the VCD file. This option should be used only when you have created multiple VCD files using the vcd files command.

  <filename> — A *.vcd* file.

- -in

  (optional) Includes only port driver changes from ports of mode IN.

- -out

  (optional) Includes only port driver changes from ports of mode OUT.

- -inout

  (optional) Includes only port driver changes from ports of mode INOUT.

- -ports

  (optional) Includes only port driver changes. Excludes internal variable or signal changes.

- -internal

  (optional) Includes only internal variable or signal changes. Excludes port driver changes.

- -r | -r -optcells

  (optional) Specifies that signal and port selection occurs recursively into subregions. If omitted, included signals and ports are limited to the current region. When -r is used with -optcells it allows Verilog optimized cell ports to be visible when using wildcards. By default Verilog optimized cell ports are not selected even if they match the specified wildcard pattern.

- <object_name> ...

  (required) Specifies the Verilog or VHDL object or objects to add to the VCD file. Multiple objects may be specified by separating names with spaces. Wildcards are accepted. Must be specified as the final argument to the vcd add command.

Related Topics

- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd checkpoint

This command dumps the current values of all VCD variables to the specified VCD file. While simulating, only value changes are dumped.

Related Verilog tasks: $dumpall, $fdumpall

Syntax

vcd checkpoint [<filename>]

Arguments

- <filename>

  (optional) Specifies the name of the VCD file. If omitted the command is executed on the file designated by the vcd file command or *dump.vcd* if vcd file was not invoked.

Related Topics

- vcd add
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd comment

This command inserts the specified comment in the specified VCD file.

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

vcd comment <comment string> [<filename>]

Arguments

- <comment string>

  (required) Comment to be included in the VCD file. Must be enclosed by double quotation marks or curly braces. Must be specified as the first argument to the vcd comment command.

- <filename>

  (optional) Specifies the name of the VCD file. If omitted the command is executed on the file designated by the vcd file command or *dump.vcd* if vcd file was not invoked.

Related Topics

- vcd add
- vcd checkpoint
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd dumpports

This command creates a VCD file that includes port driver data.

By default all port driver changes are captured in the file. You can filter the output using arguments detailed below. Related Verilog task: $dumpports

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

vcd dumpports [-compress] [-direction] [-file <filename>] [-force_direction] [-in] [-out] [-inout]
[-no_strength_range] [-unique] [-vcdstim] **<object_name> ...**

Arguments

- -compress

  (optional) Produces a compressed VCD file. Questa SIM uses the gzip compression algorithm. It is not necessary to specify -compress if you specify a *.gz* extension with the -file <filename> argument

- -direction

  (optional) Includes driver direction data in the VCD file.

- -file <filename>

  (optional) Creates a VCD file. Defaults to the current working directory and the filename *dumpports.vcd*. Multiple filenames can be opened during a single simulation.

  <filename> — Specifies a filename. When specified with a *.gz* extension, the file is compressed.

- -force_direction

  (optional) Causes vcd dumpports to use the specified port direction (instead of driver location) to determine whether the value being dumped is input or output. This argument overrides the default use of the location of drivers on the net to determine port direction (this is because Verilog port direction is not enforced by the language or by Questa SIM).

- -in

  (optional) Includes ports of mode IN.

- -out

  (optional) Includes ports of mode OUT.

- -inout

  (optional) Includes ports of mode INOUT.

- -no_strength_range

  (optional) Ignores strength ranges when resolving driver values. This argument is an extension to the IEEE 1364 specification. Refer to Resolving Values for additional information.

- -unique

  (optional) Generates unique VCD variable names for ports even if those ports are connected to the same collapsed net.

- -vcdstim

  (optional) Ensures that port name order in the VCD file matches the declaration order in the instance module or entity declaration. Refer to Port Order Issues for further information.

- <object_name> ...

  (required) Specifies one or more HDL objects to add to the VCD file. You can specify multiple objects by separating names with spaces. Wildcards are accepted. Must be specified as the final argument to the vcd dumpports command.

Examples

- Create a VCD file named *counter.vcd* of all IN ports in the region /test_design/dut/.

  ```
  vcd dumpports -in -file counter.vcd /test_design/dut/*
  ```

- These two commands resimulate a design from a VCD file. Refer to Simulating with Input Values from a VCD File for further details.

  ```
  vcd dumpports -file addern.vcd /testbench/uut/*
  vsim -vcdstim addern.vcd addern -gn=8 -do "add wave /*; run 1000"
  ```

- This series of commands creates VCD files for the instances *proc* and *cache* and then re-simulates the design using the VCD files in place of the instance source files. Refer to Replacing Instances with Output Values from a VCD File for more information.

  ```
  vcd dumpports -vcdstim -file proc.vcd /top/p/*
  vcd dumpports -vcdstim -file cache.vcd /top/c/*
  run 1000

  vsim top -vcdstim /top/p=proc.vcd -vcdstim /top/c=cache.vcd
  ```

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd dumpportsall

This command creates a checkpoint in the VCD file which shows the value of all selected ports at that time in the simulation, regardless of whether the port values have changed since the last timestep.

Related Verilog task: $dumpportsall

Syntax

vcd dumpportsall [<filename>]

Arguments

- <filename>

  (optional) Specifies the name of the VCD file. If omitted the command is executed on all open VCD files.

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd dumpportsflush

This command flushes the contents of the VCD file buffer to the specified VCD file.

Related Verilog task: $dumpportsflush

Syntax

vcd dumpportsflush [<filename>]

Arguments

- <filename>

  (optional) Specifies the name of the VCD file. If omitted the command is executed on all open VCD files.

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd dumpportslimit

This command specifies the maximum size of the VCD file (by default, limited to available disk space). When the size of the file exceeds the limit, a comment is appended to the file and VCD dumping is disabled.

Related Verilog task: $dumpportslimit

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

vcd dumpportslimit **<dumplimit>** [<filename>]

Arguments

- <dumplimit>

  (required) Specifies the maximum VCD file size in bytes. Must be specified as the first argument to the vcd dumpportslimit command.

- <filename>

  (optional) Specifies the name of the VCD file. If omitted the command is executed on all open VCD files.

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd dumpportsoff

This command turns off VCD dumping and records all dumped port values as x.

Related Verilog task: $dumpportsoff

Syntax

vcd dumpportsoff [<filename>]

Arguments

- <filename>

  (optional) Specifies the name of the VCD file. If omitted the command is executed on all open VCD files.

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd dumpportson

This command turns on VCD dumping and records the current values of all selected ports. This command is typically used to resume dumping after invoking vcd dumpportsoff.

Related Verilog task: $dumpportson

Syntax

vcd dumpportson [<filename>]

Arguments

- <filename>

  (optional) Specifies the name of the VCD file. If omitted the command is executed on all open VCD files.

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd file

This command specifies the filename and state mapping for the VCD file created by a vcd add command. The vcd file command is optional. If used, it must be issued before any vcd add commands.

Related Verilog task: $dumpfile

Syntax

vcd file [-dumpports] [-direction] [<filename>] [-map <mapping pairs>] [-no_strength_range] [-nomap] [-unique]

Arguments

- -dumpports

  (optional) Capture detailed port driver data for Verilog ports and VHDL std_logic ports. This option works only on ports, and any subsequent vcd add command will accept only qualifying ports (silently ignoring all other specified objects).

- -direction

  (optional) Includes driver direction data in the VCD file.

- <filename>

  (optional) Specifies the name of the VCD file that is created where the default is *dump.vcd*.

- -map <mapping pairs>

  (optional) Overrides the default mappings. Affects only VHDL signals of type std_logic.

  <mapping pairs> — Specified as a list of character pairs. The first character in a pair must be one of the std_logic characters UX01ZWLH- and the second character is the character you wish to be recorded in the VCD file. The Tcl convention for command strings that include spaces is to enclose them in quotation marks (" "). For example, to map L and H to z:

  vcd file -map "L z H z"

- -no_strength_range

  (optional) Ignores strength ranges when resolving driver values. This argument is an extension to the IEEE 1364 specification. Refer to Resolving Values for additional information.

- -nomap

  (optional) Affects only VHDL signals of type std_logic. It specifies that the values recorded in the VCD file shall use the std_logic enumeration characters of UX01ZWLH-. This option results in a non-standard VCD file because VCD values are limited to the four state character set of x01z. By default, the std_logic characters are mapped as follows.

VHDL	VCD	VHDL	VCD
U	x	W	x
X	x	L	0
0	0	H	1
1	1	-	x
Z	z		

- -unique

  (optional) Generates unique VCD variable names for ports even if those ports are connected to the same collapsed net.

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd files

This command specifies filenames and state mapping for VCD files created by the vcd add command. The vcd files command is optional. If used, it must be issued before any vcd add commands.

Related Verilog task: $fdumpfile

Syntax

vcd files [-compress] [-direction] <filename> [-map <mapping pairs>] [-no_strength_range] [-nomap] [-unique]

Arguments

- -compress

  (optional) Produces a compressed VCD file. Questa SIM uses the gzip compression algorithm. If you specify a *.gz* extension on the -file <filename> argument, Questa SIM compresses the file even if you don't use the -compress argument.

- -direction

  (optional) Includes driver direction data in the VCD file.

- <filename>

  (required) Specifies the name of a VCD file to create. Multiple files can be opened during a single simulation; however, you can create only one file at a time. If you want to create multiple files, invoke vcd files multiple times.

- -map <mapping pairs>

  (optional) Overrides the default mappings. Affects only VHDL signals of type std_logic.

  <mapping pairs> — Specified as a list of character pairs. The first character in a pair must be one of the std_logic characters UX01ZWLH- and the second character is the character you wish to be recorded in the VCD file. The Tcl convention for command strings that include spaces is to enclose them in quotation marks (" "). For example, to map L and H to z:

  vcd file -map "L z H z"

- -no_strength_range

  (optional) Ignores strength ranges when resolving driver values. This argument is an extension to the IEEE 1364 specification. Refer to "Resolving Values" for additional information.

- -nomap

  (optional) Affects only VHDL signals of type std_logic. It specifies that the values recorded in the VCD file shall use the std_logic enumeration characters of UX01ZWLH-. This option

results in a non-standard VCD file because VCD values are limited to the four state character set of x01z. By default, the std_logic characters are mapped as follows.

VHDL	VCD	VHDL	VCD
U	x	W	x
X	x	L	0
0	0	H	1
1	1	-	x
Z	z		

- -unique

  (optional) Generates unique VCD variable names for ports even if those ports are connected to the same collapsed net.

Examples

The following example shows how to "mask" outputs from a VCD file until a certain time after the start of the simulation. The example uses two vcd files commands and the vcd on and vcd off commands to accomplish this task.

```
vcd files in_inout.vcd
vcd files output.vcd
vcd add -in -inout -file in_inout.vcd /*
vcd add -out -file output.vcd /*
vcd off output.vcd
run 1us
vcd on output.vcd
run -all
```

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd flush
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd flush

This command flushes the contents of the VCD file buffer to the specified VCD file. This command is useful if you want to create a complete VCD file without ending your current simulation.

Related Verilog tasks: $dumpflush, $fdumpflush

Syntax

vcd flush [<filename>]

Arguments

- <filename>

  (optional) Specifies the name of the VCD file. If omitted, the command is executed on the file designated by the vcd file command or *dump.vcd* if vcd file was not invoked.

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd limit
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd limit

This command specifies the maximum size of a VCD file (by default, limited to available disk space).

When the size of the file exceeds the limit, a comment is appended to the file and VCD dumping is disabled.

Related Verilog tasks: $dumplimit, $fdumplimit

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

vcd limit <filesize> [<filename>]

Arguments

- <filesize>

  (Required) Specifies the maximum VCD file size, in bytes. The numerical value of <filesize> can only be a whole number. Must be specified as the first argument to the vcd limit command.

  You can specify a unit of Kb, Mb, or Gb with the numerical value (units are case insensitive). Do not insert a space between the numerical value and the unit (for example, 400Mb, not 400 Mb).

- <filename>

  (Optional) Specifies the name of the VCD file. If omitted, the command is executed on the file designated by the vcd file command or *dump.vcd* if vcd file was not invoked.

Examples

- Specify a maximum VCD file size of 6 gigabytes and a VCD file named my_vcd_file.vcd.

  vcd limit 6gb my_vcd_file.vcd

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd off
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd off

This command turns off VCD dumping to the specified file and records all VCD variable values as x.

Related Verilog tasks: $dumpoff, $fdumpoff

Syntax

vcd off [<filename>]

Arguments

- <filename>

  (optional) Specifies the name of the VCD file. If omitted, the command is executed on the file designated by the vcd file command or *dump.vcd* if vcd file was not invoked.

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd on
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd on

This command turns on VCD dumping to the specified file and records the current values of all VCD variables.

By default, vcd on is automatically performed at the end of the simulation time that the vcd add command performed.

Related Verilog tasks: $dumpon, $fdumpon

Syntax

vcd on [<filename>]

Arguments

- <filename>

  (optional) Specifies the name of the VCD file. If omitted the command is executed on the file designated by the vcd file command or *dump.vcd* if vcd file was not invoked.

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd2wlf
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcd2wlf

This command is a utility that translates a VCD (Value Change Dump) file into a WLF file that you can display in Questa SIM using the vsim -view argument. This command only works on VCD files containing positive time values.

Description

The vcd2wlf command functions as simple one-pass converter. If you are defining a bus in a VCD file, you must specify all bus bits before the next $scope or $upscope statement appears in the file. The best way to ensure that bits get converted together as a bus is to declare them on consecutive lines.

For example:

```
Line 21 : $var wire 1 $ in [2] $end
Line 22 : $var wire 1 $u in [1] $end
Line 23 : $var wire 1 # in [0] $end
```

Arguments to this command are order dependent. Please read the argument descriptions for more information.

Syntax

vcd2wlf [-splitio] [-splitio_in_ext <extension>] [-splitio_out_ext <extension>] [-nocase] {<vcd filename> | – } <wlf filename>

Arguments

- -splitio

  (optional) Specifies that extended VCD port values are to be split into their corresponding input and output components by creating two signals instead of just one in the resulting *.wlf* file. By default the new input-component signal keeps the same name as the original port name while the output-component name is the original name with "__o" appended to it.

- -splitio_in_ext <extension>

  (optional) Adds an extension to input-component signal names created by using -splitio.

  <extension> — Specifies a string.

- -splitio_out_ext <extension>

  (optional) Adds an extension to output-component signal names created by using -splitio.

  <extension> — Specifies a string.

- -nocase

  (optional) Converts all alphabetic identifiers to lowercase.

- {<vcd filename> | – }

  (required) Specifies the name of the VCD file, or standard input (-), you want to translate into a WLF file. Must be specified immediately preceding the <wlf filename> argument to the vcd2wlf command.

- <wlf filename>

  (required) Specifies the name of the output WLF file. Must be specified as the final argument to the vcd2wlf command.

Examples

- Concatenate *my.vcd* file and pipe standard input to vcd2wlf and save output to *my.wlf* file.

  ```
  cat my.vcd | vcd2wlf - my.wlf
  ```

- Redirect input from the file *my.vcd* file to vcd2wlf and save the output to *my.wlf* file.

  ```
  vcd2wlf - my.wlf  <my.vcd
  ```

Related Topics

- vcd add
- vcd checkpoint
- vcd comment
- vcd dumpports
- vcd dumpportsall
- vcd dumpportsflush
- vcd dumpportslimit
- vcd dumpportsoff
- vcd dumpportson
- vcd file
- vcd files
- vcd flush
- vcd limit
- vcd off
- vcd on
- DumpportsCollapse
- Value Change Dump (VCD) Files
- Verilog tasks are documented in the Language Reference Manual IEEE 1364 standard.

vcom

The vcom command compiles VHDL source code into a specified working library (or to the work library by default).

You can invoke vcom either from within Questa SIM or from the command prompt of your operating system. You can invoke this command during simulation.

Compiled libraries are dependent on the major version of Questa SIM. When moving between major versions, you must refresh compiled libraries using the -refresh argument to vcom. This is not required for minor versions (letter releases).

All arguments to the vcom command are case-sensitive. For example, -WORK and -work are not equivalent.

This command provides additional information with the -help or -h switch.

Syntax

vcom [options] <filename> [<filename> ...]

[options]:

[-32 | -64] [-87 | -93 | -2002 | -2008]

[+acc[=<spec>][+<entity>[(architecture)]]]

Note

Use of the +acc argument with vcom is not recommended for the following reasons:

- It may reduce simulation speed.
- You should use the access debug arguments for the vopt command instead. This will simplify compilation and optimization flows and make them easier to maintain.
- The +acc argument and its values have been replaced by more specific arguments to the vcom and vopt commands. The +acc argument is supported but deprecated.

Refer to Using vcom for Access Control for Visibility During Optimization for information on replacements for vcom +acc.

[-addpragmaprefix <prefix>] [-allowProtectedBeforeBody] [-amsstd | -noamsstd]

[-bindAtCompile] [-bindAtLoad]

[-check_synthesis] [-constimmedassert | -noconstimmedassert] [+cover[=<spec>]]
[-coverclkoptbuiltins | -nocoverclkoptbuiltins]
[-coverdeglitch {“<n> <time_unit>”}] [-coverenhanced]
[-coverexcludedefault | -nocoverexcludedefault]
[-coverfec | -nocoverfec] [-coverrec | -nocoverrec]

[-coveropt <opt_level>] [-coverreportcancelled]
[-coversub | -nocoversub] [-coverudp | [-nocoverudp]

[-debugVA] [-defercheck] [-deferSubpgmCheck | -noDeferSubpgmCheck]

[-error <msg_number>[,<msg_number>,…]] [-explicit] [-extendedtogglemode 1|2|3]

[(-F | -file | -f) <filename>] [-fatal <msg_number>[,<msg_number>,…]]
[-fecudpeffort {1 | 2 | 3}] [-force_refresh <primary> [<secondary>]]
[-fsmimplicittrans | -nofsmimplicittrans] [-fsmmultitrans]
[-fsmresettrans | -nofsmresettrans] [-fsmsingle | -nofsmsingle]
[-fsmverbose [b | t | w]]

[-gen_xml <design_unit> <filename>]

[-ignoredefaultbinding] [-ignorepragmaprefix <prefix>] [ignoreStandardRealVector]
[-ignorevitalerrors] [-initoutcompositeparam | -noinitoutcompositeparam]

[-just abcep]

[-logfile <filename> | -l <filename>] [-line <number>] [-lint] [-lower]
[-lrmconfigvis]

[-mixedsvvh [b | l | r][i]] [-modelsimini <path/modelsim.ini>]
[-msglimit <msg_number>[,<msg_number>,…]]

[-no1164] [-noaccel <package_name>] [-nocasestaticerror] [-nocheck]
[-nocoverrespecthandl] [-nocreatelib] [-nodbgsym] [-nofprangecheck]
[-noFunctionInline] [-noindexcheck] [-nologo] [-nonstddriverinit]
[-noothersstaticerror] [-note <msg_number> [,<msg_number>, …]] [-novital]
[-novitalcheck] [-novopt] [-nowarn <category_number>] [-nocovershort]
[-nodebug[=ports]] [-nopsl]

[-O0 | -O1 | -O4 | -O5] [-oldconfigvis] [-optionset <optionset_name>]
[-outf <filename>]

[-pedanticerrors] [-performdefaultbinding] [-preserve] [-[w]prof=<filename>]
[-proftick=<integer>] [+protect [=<filename>]] [-pslext] [-pslfile <filename>]

[-quiet]

[-rangecheck | -norangecheck] [-refresh <primary> [<secondary>]]

[-s] [-separateConfigLibrary] [-showsubprograms | -noshowsubprograms] [-skip abcep]
[-skipsynthoffregion] [-smartdbgsym] [-source]
[-stats [=[+ | -]<feature>[,[+ | -]<mode>]]]
[-suppress <msg_number>[,<msg_number>,…]]

[-togglecountlimit <int>] [-togglewidthlimit <int>] [-toggleportsonly]

[-version] [-vitalmemorycheck] [-vmake] [-vopt | -novopt]

[-warning <msg_number>[,<msg_number>,…]] [-warning error] [-work
<library_name>]

Using vcom for Access Control for Visibility During Optimization

The following optional arguments are replacements for the +acc argument and its values. Note that the +acc argument is still supported for backward compatibility, but its use is deprecated in favor of the vcom arguments listed below.

vcom [-assertdebug [=[-]<accesscodes>]] [-fsmdebug [=[-]<accesscodes>]] [-floatgenerics] [-linedebug [=[-]<accesscodes>]] <filename> [<filename> ...]

Arguments

- -32 | -64

  Specifies whether vcom uses the 32- or 64-bit executable, where -32 is the default.

  These options apply only to the supported Linux operating systems.

  These options override the MTI_VCO_MODE environment variable, which applies only to executables used from the <*install_dir*>*/bin/* directory. Therefore, these options are ignored if you run vcom from an <*install_dir*>/<*platform*>/ directory.

  You can only specify these options on the command line, therefore they are not recognized as part of a file used with the -f switch.

- -87 | -93 | -2002 | -2008

  (optional) Specifies which LRM-specific compiler to use. You can also control this behavior with the VHDL93 variable in the *modelsim.ini* file. Refer to "Differences Between Versions of VHDL" for more information.

  -87 — Enables support for VHDL 1076-1987.

  -93 — Enables support for VHDL 1076-1993.

  -2002 — Enables support for VHDL 1076-2002. (default)

  -2008 — Enables support for VHDL 1076-2008.

- +acc[=<spec>][+<entity>[(architecture)]]

  (optional) Enable debug command access to objects indicated by <spec> when optimizing a design. The effect of this argument is limited to only those design units being compiled in the current vcom session.

  <spec> — A string of one or more of the following characters:

  f — Enable access to finite state machines

  v — Enable access to variables, constants, and aliases in processes that would otherwise be merged due to optimizations. Disables an optimization that automatically converts variables to constants.

  If <spec> is omitted, access is enabled for all objects.

  <+entity> — specify the VHDL design regions in which to allow access. All architectures for <entity> are enabled for access.

<architecture> — specify the VHDL design regions in which to allow access. If (<architecture>) is not specified, then all architectures of a given <entity> are enabled for access. May be optionally followed by "." to indicate all children of the module. Supports enum values as the index for blocks created by VHDL for generate statements.

- -addpragmaprefix <prefix>

  (optional) Enables recognition of pragmas with a user specified prefix. If this argument is not specified, pragmas are treated as comments.

  All regular synthesis and coverage pragmas are honored.

  <prefix> — Specifies a user defined string where the default is no sting, indicated by quotation marks.

  You may also set this with the AddPragmaPrefix variable in the vcom section of the *modelsim.ini* file.

- -allowProtectedBeforeBody

  (optional) Allows a variable of a protected type to be created prior to declaring the body.

- -amsstd | -noamsstd

  (optional) Specifies whether vcom adds the declaration of REAL_VECTOR to the STANDARD package. This is useful for designers using VHDL-AMS to test digital parts of their model.

  -amsstd — REAL_VECTOR is included in STANDARD.

  -noamsstd — REAL_VECTOR is not included in STANDARD (default).

  You can also control this with the AmsStandard variable or the MGC_AMS_HOME environment variable.

- -assertdebug [=[-]<accesscodes>]

  (optional) Allow to debug SVA/PSL objects when used with vsim -assertdebug.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

- -bindAtCompile

  (optional) Forces Questa SIM to perform default binding at compile time rather than at load time. Refer to “Default Binding” for more information. You can change the permanent default by editing the BindAtCompile variable in the *modelsim.ini*.

- -bindAtLoad

  (optional) Forces Questa SIM to perform default binding at load time rather than at compile time. (Default)

- -check_synthesis

  (optional) Turns on limited synthesis rule compliance checking. Specifically, it checks to see that signals read by a process are in the sensitivity list. The checks understand only combinational logic, not clocked logic. Edit the CheckSynthesis variable in the *modelsim.ini* file to set a permanent default.

- -constimmedassert

  (optional) Displays immediate assertions with constant expressions in the GUI, in reports, and in the UCDB. By default, immediate assertions with constant expressions are displayed in the GUI, in reports, and in the UCDB. Use this switch only if the -noconstimmedassert switch has been used previously, or if the ShowConstantImmediateAsserts variable in the vcom section of the *modelsim.ini* file is set to 0 (off).

- -noconstimmedassert

  (optional) Turns off the display of immediate assertions with constant expressions in the GUI, in reports, and in the UCDB. By default, immediate assertions with constant expressions are displayed. You may also set the ShowConstantImmediateAsserts variable in the vcom section of the *modelsim.ini* file to 0 (off).

- +cover[=<spec>]

  (optional) Enables various coverage statistics collection on all design units compiled in the current compiler run.

  Consider using the +cover or +nocover arguments to vopt instead, which you can use to specify the precise design units and regions to be instrumented (or uninstrumented) for coverage.

  The +cover argument with no "=<spec>" designation is equivalent to "+cover=bcesft".

  <spec> — A string of one or more of the following characters, spaces not allowed:

  b — Collect branch statistics.

  c — Collect condition statistics. Collects only FEC statistics, unless -coverudp is specified.

  e — Collect expression statistics, Collects only FEC statistics, unless -coverudp is specified.

  s — Collect statement statistics.

  t — Collect toggle statistics. Overridden if ‘x’ is specified elsewhere.

  x — Collect extended toggle statistics (Refer to “Toggle Coverage” for details). This takes precedence, if ‘t’ is specified elsewhere.

  f — Collect Finite State Machine statistics.

Refer to -coveropt <opt_level> argument to override the default level of optimization for coverage for a particular compilation run.

Refer to "Rules for Applying Coverage with cover and nocover" for information on how potentially conflicting commands are resolved.

- -coverenhanced

  (optional) Enables non-critical functionality which might change the appearance or content of coverage metrics. This argument only has an effect in letter releases (10.0a, 10.0b, and so on). In major releases (10.0, 10.1, and so on), all coverage enhancements present in previous letter release streams are enabled by default, and -coverenhanced is no longer necessary to enable these enhancements. Bug fixes important to the correctness of coverage numbers are always enabled by default, with no need for -coverenhanced. Since the exact nature of -coverenhanced varies from release to release, the details of the enhancements it enables are present in the product release notes rather than in the Command Reference. For these details, search the release notes using the string "coverenhanced".

- -coverexcludedefault

  (optional) Excludes VHDL code coverage data collection for the OTHERS branch in both Case statements and Selected Signal Assignment statements. All forms of code coverage collection for statements contained in the OTHERS branch will be excluded.

- -coverfec

  (optional) Enables focused expression coverage (FEC) for coverage collection. By default, FEC coverage statistics are disabled for collection. You can customize the default behavior with the CoverFEC variable in the *modelsim.ini* file.

- -coverclkoptbuiltins | -nocoverclkoptbuiltins

  (optional) Enables (default) / Disables VHDL clock optimization builtins for code coverage. By default, Allfalse branches are designated in the coverage report with "ECOP" (if not hit, "E-hit" if hit), indicating that it was excluded because of the "clock optimization". This optimization can be turned off using the -noclkoptbuiltins. You can customize the default behavior with the CoverClkOptBuiltins variable in the *modelsim.ini* file. For more information, see "Missing Branches in VHDL and Clock Optimizations".

- -coverdeglitch {"<n> <time_unit>"}

  (optional) Enables deglitching of statement, branch, condition and expression coverage in combinational unclocked process/always blocks, concurrent (VHDL) and continuous (SV) assignments, where:

  [n] — A string specifying the number of time units.

  <time_unit> — (required if "n" is anything other than "0") fs, ps, ns, us, ms, or sec.

  Quotes ("") are required if a space exists between the two arguments (for example, 2ps or "2 ps").

  If a process or a continuous assignment is evaluated more than once during any period of length <period>, only the last execution during that period is added into the coverage data

for that process/continuous assignment. If a deglitch period of zero is specified, only the last delta cycle pass is counted toward the coverage data. For a new pass to be counted, it must occur at a time greater than the previous pass plus the deglitch period. You can customize the default behavior with the CoverDeglitchOn and CoverDeglitchPeriod variables in the *modelsim.ini* file.

- -coveropt <opt_level>

  (optional) Overrides the default level of optimization for the current run only.

  <opt_level> specifies one of the following optimization levels:

  1 — Turns off all optimizations that affect coverage reports.

  2 — Allows optimizations that provide large performance improvements by invoking sequential processes only when the data changes. This setting may result in major reductions in coverage counts.

  3 — (default) Allows all optimizations in 2, and allows optimizations that may remove some statements. Also allows constant propagation and VHDL subprogram inlining.

  4 — Allows all optimizations in 2 and 3, and allows optimizations that may remove major regions of code by changing assignments to built-ins or removing unused signals. It also changes Verilog gates to continuous assignments and optimizes Verilog expressions. Allows VHDL subprogram inlining. Allows VHDL flip-flop recognition.

  5— Allows all optimizations in 2-4 and activates code coverage for Verilog merged always blocks, merged initialization blocks, merged final blocks, and merged if statements.

  You can edit the CoverOpt variable in the *modelsim.ini* file to change the default.

- -coverrec

  (optional) Enables rapid expression coverage (REC) for coverage collection. By default, REC coverage statistics are enabled for collection. You can customize the default behavior with the CoverREC variable in the *modelsim.ini* file.

- -coverreportcancelled

  (optional) Enables code coverage reporting of branch conditions that have been optimized away due to a static or null condition. The line of code is labeled EA in the hits column of the Source Window and EBCS in the hits column of a Coverage Report. You can also set this with the CoverReportCancelled *modelsim.ini* variable.

- -coversub

  (optional) Re-enables code coverage data collection in VHDL subprograms previously disabled with -nocoversub. By default code coverage data is collected for VHDL subprograms. Edit the CoverSub variable in the *modelsim.ini* file to set a permanent default.

- -coverudp

  (optional) Enables UDP coverage for coverage collection. By default, FEC coverage statistics are enabled for collection. You can customize the default behavior with the CoverUDP variable in the *modelsim.ini* file.

- -debugVA

  (optional) Prints a confirmation if a VITAL cell was optimized, or an explanation of why it was not, during VITAL level-1 acceleration.

- -defercheck

  (optional) Defers index checks until run time.

- -deferSubpgmCheck

  (optional) Forces the compiler to report array indexing and length errors as warnings (instead of as errors) when encountered within subprograms. Subprograms with indexing and length errors that are invoked during simulation cause the simulator to report errors, which can potentially slow down simulation because of additional checking.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information.

  <msg_number> — A number identifying the message. Multiple message numbers are specified as a comma separated list.

- -explicit

  (optional) Directs the compiler to resolve ambiguous function overloading by favoring the explicit function definition over the implicit function definition. Strictly speaking, this behavior does not match the VHDL standard. However, the majority of EDA tools choose explicit operators over implicit operators. Using this switch makes Questa SIM compatible with common industry practice.

- -extendedtogglemode 1|2|3

  (optional) Changes the level of support for extended toggles. The levels of support are:

  1 — 0L->1H & 1H->0L & any one 'Z' transition (to/from 'Z')

  2 — 0L->1H & 1H->0L & one transition to 'Z' & one transition from 'Z'

  3 — 0L->1H & 1H->0L & all 'Z' transitions

  Edit the ExtendedToggleMode variable in the *modelsim.ini* file to set a permanent default.

- (-F | -file | -f) <filename>

  (optional) -f, -file and -F: each specifies an argument file with more command-line arguments, allowing complex argument strings to be reused without retyping. Nesting of -F, -f and -file commands is allowed. Allows gzipped input files.

With -F only: relative file names and paths within the arguments file <filename> are prefixed with the path of the arguments file when lookup with relative path fails. Refer to the section "Argument Files" for more information.

- -fatal <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "fatal." Edit the fatal variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

  <msg_number> — A number identifying the message. Multiple message numbers are specified as a comma separated list.

- -fecudpeffort {1 | 2 | 3}

  (optional) The recommended option to set the level of effort for FEC or UDP. Levels are:

  1 — (default) (low) Only small expressions or conditions are considered for coverage.

  2 — (medium) Bigger expressions and conditions considered for coverage.

  3 — (high) Very large expressions and conditions considered for coverage.

  Increasing the effort level includes more expressions for coverage, but also increases the compile time. You can also enable this by setting the FecUdpEffort variable in the *modelsim.ini* file.

- -floatgenerics

  (optional) Do not hold generic values constant during optimization. This enables use of the vsim -g/G arguments on the affected generics.

- -force_refresh <primary> [<secondary>]

  (optional) Forces the refresh of all specified design units. By default, the work library is updated; use -work <library_name>, in conjunction with -force_refresh, to update a different library (for example, vcom -work <your_lib_name> -force_refresh).

  <primary> [<secondary>] — Specifies the entity, package, configuration, or module to be deleted.

  This option is not supported for SystemC modules.

  - If <primary> is an entity — only that entity, no related architectures, is refreshed.
  - If <primary> is a package — the only legal value of <secondary> is "body", and only the package is refreshed.
  - If you specify both <primary> and <secondary> — Only the <secondary> architecture is updated, not the entity.

  When the compiler refreshes a design unit, it checks each dependency to ensure its source has not been changed and recompiled. Sometimes the dependency checking algorithm changes from release to release. This can lead to false errors during the integrity checks performed by the -refresh argument. An example of such a message follows:

```
** Error: (vsim-13) Recompile /u/test/dware/dware_61e_beta.dwpackages
because /home/users/questasim/../synopsys.attributes has changed.
```

The -force_refresh argument forces the refresh of the design unit, overriding any dependency checking errors encountered by the -refresh argument.

A more conservative approach to working around -refresh dependency checks is to recompile the source code, if it is available.

You cannot specify the <filename> argument when specifying this argument.

- -fsmdebug [=[-]<accesscodes>]

  (optional) Allow access to finite state machines for debugging. Formerly, vcom +acc=f.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

- -fsmimplicittrans | -nofsmimplicittrans

  (optional) Toggles recognition of implied same state transitions, which is off by default (-nofsmimplicittrans).

- -fsmmultitrans

  (optional) Enables detection and reporting of multi-state transitions when used with the +cover=f argument for vcom or vopt. Another term for this is FSM sequence coverage.

- -fsmresettrans | -nofsmresettrans

  (optional) Toggles recognition of sychronous or asynchronous reset transitions, and is on by default (-fsmresettrans).

  This includes/excludes reset transitions in coverage results.

- -fsmsingle | -nofsmsingle

  (optional) Toggles the recognition of VHDL FSMs where the current state variable of type std_logic, bit, boolean, or single-bit std_logic_vector/bit_vector and Verilog single-bit FSMs.

- -fsmverbose [b | t | w]

  (optional) Provides information about FSMs detected, including state reachability analysis.

  b — displays only basic information.

  t — displays a transition table in addition to the basic information.

  w — displays any warning messages in addition to the basic information.

  When you do not specify an argument, this switch reports all information similar to:

```
# ** Note: (vcom-1947)   FSM RECOGNITION INFO
#      Fsm detected in : ../fpu/rtl/vhdl/serial_mul.vhd
#      Current State Variable : s_state :
../fpu/rtl/vhdl/serial_mul.vhd(76)
#      Clock : clk_i
#      Reset States are: { waiting , busy }
#      State Set is : { busy , waiting }
#      Transition table is
#         ----------------------------------------
#         busy      =>   waiting Line : (114 => 114)
#         busy      =>   busy    Line : (111 => 111)
#         waiting   =>   waiting Line : (120 => 120) (114 => 114)
#         waiting   =>   busy    Line : (111 => 111)
#         ----------------------------------------
```

When you do not specify this switch, you will receive a message similar to:

```
# ** Note: (vcom-143) Detected '1' FSM/s in design unit
'serial_mul.rtl'.
```

- -gen_xml <design_unit> <filename>

  (optional) Produces an XML-tagged file containing the interface definition of the specified entity.

  <design_unit> — The name of an entity or design unit in the Work library. Wildcards and multiple design unit names are not allowed.

  <filename> — A user-specified name for the file.

  For example:

  This option requires a two-step process where you must:

  1) compile <filename> into a library with vcom (without -gen_xml) then

  2) execute vcom with the -gen_xml switch.

```
vlib work
vcom counter.vhd
vcom -gen_xml counter counter.xml
```

- -ignoredefaultbinding

  (optional) Instructs the compiler not to generate a default binding during compilation. You must explicitly bind all components in the design through either configuration specifications or configurations. If an explicit binding is not fully specified, defaults for the architecture, port maps, and generic maps will be used as needed. Refer to Disabling Default Binding for more information. Edit the RequireConfigForAllDefaultBinding *modelsim.ini* variable to set a permanent default.

- -ignorepragmaprefix <prefix>

  (optional) Directs vcom to ignore pragmas with the specified prefixname. All affected pragmas will be treated as regular comments. Edit the IgnorePragmaPrefix *modelsim.ini* variable to set a permanent default.

  <prefix> — Specifies a user defined string.

- ignoreStandardRealVector

  (optional) Instructs ModelSim to ignore the REAL_VECTOR declaration in package STANDARD when compiling with vcom -2008. Edit the ignoreStandardRealVector *modelsim.ini* variable to set a permanent default. For more information refer to the REAL_VECTOR section in **Help > Technotes > vhdl2008migration**.

- -ignorevitalerrors

  (optional) Directs the compiler to ignore VITAL compliance errors. The compiler still reports that VITAL errors exist, but it will not stop the compilation. You should exercise caution in using this switch; as part of accelerating VITAL packages, we assume that compliance checking has passed.

- -initoutcompositeparam

  (optional) Causes initialization of subprogram parameters for array and record types when the subprogram is executed in designs compiled with LRM 1076-2002 and earlier. This argument forces the output parameters to their default initial ("left") values when entering a subprogram. By default, -initoutcompositeparam is enabled for designs compiled with vcom -2008 and later. You can also enable this by setting the InitiOutCompositeParam variable to 1 in the *modelsim.ini* file.

- -noinitoutcompositeparam

  (optional) Disables initialization of subprogram parameters for array and record types when the subprogram is executed in designs compiled with LRM 1076-2002 and earlier. By default, designes compiled with LRM 1076-2008 and later do not initialize subprogram parameters for array and record types when the subprogram is executed. You can also disable initialization of subprogram parameters for array and record types by setting the InitiOutCompositeParam variable to 2 in the *modelsim.ini* file.

- -just abcep

  (optional) Directs the compiler to include only the following:

  a — architectures

  b — bodies

  c — configurations

  e — entities

  p — packages

  Any combination in any order can be used, but you must specify at least one choice if you use this switch.

- -logfile <filename> | -l <filename>

  (optional) Generates a log file of the compile.

  -logfile <filename> — Saves transcript data to <filename>. Can be abbreviated to -l <filename>. Overrides the default transcript file creation set with the the

TranscriptFile or BatchTranscriptFile *modelsim.ini* variables. You can also specify “stdout” or “stderr” for <filename>.

- -line <number>

  (optional) Starts the compiler on the specified line in the VHDL source file. By default, the compiler starts at the beginning of the file.

  <number> —

- -linedebug [=[-]<accesscodes>]

  (optional) Allow line debugging.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

- -lint

  (optional) Performs additional static checks on case statement rules and enables warning messages for the following situations:

  - The result of the built-in concatenation operator ("&") is the actual for a subprogram formal parameter of an unconstrained array type.
  - If you specify the -BindAtCompile switch with vcom, the entity to which a component instantiation is bound has a port that is not on the component, and for which there is no error otherwise.
  - A direct recursive subprogram call.
  - In cases involving class SIGNAL formal parameters, as described in the IEEE Standard VHDL Language Reference Manual entitled "Signal parameters". This check only applies to designs compiled using -87. If you compile using -93, it would be flagged as a warning or error, even without the -lint argument. Can also be enabled using the Show_Lint variable in the *modelsim.ini* file.

- -lower

  (optional) Forces vcom to convert uppercase letters in object identifiers to lowercase. You can also enable this by setting the PreserveCase variable to 0 in the *modelsim.ini* file.

- -lrmconfigvis

  (optional, default) Forces vcom to use visibility rules that comply with the Language Reference Manual when processing VHDL configurations. Refer to vcom -oldconfigvis or the oldVHDLConfigurationVisibility variable in the *modelsim.ini* file for more information.

- -mixedsvvh [b | l | r][i]

  (optional) Facilitates using VHDL packages at the SystemVerilog-VHDL boundary of a mixed-language design. When you compile a VHDL package with -mixedsvvh, the package can be included in a SystemVerilog design as if it were defined in SystemVerilog itself.

  Executing -mixedsvvh without arguments compiles VHDL vectors in the following ways:

  - VHDL bit_vectors are treated as SystemVerilog bit vectors.
  - VHDL std_logic_vectors, std_ulogic_vectors, and vl_logic_vectors are treated as SystemVerilog logic vectors.

    b — treats all scalars and vectors in the package as SystemVerilog bit type

    l — treats all scalars and vectors in the package as SystemVerilog logic type

    r — treats all scalars and vectors in the package as SystemVerilog reg type

    i — ignores the range specified with VHDL integer types. Can be specified together with b, l, or r, spaces are not allowed between arguments.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable.

  <path/modelsim.ini> — Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -msglimit <msg_number>[,<msg_number>,…]

  (optional) Limits the number of iterations of the specified message(s) to five then suppresses all new instances. Refer to "Suppression of Warning Messages" for more information.

  <msg_number>[,<msg_number>,…] — Specifies the message number(s) to limit to five iterations. Multiple messages are specified as a comma-separated list.

- -no1164

  (optional) Causes the source files to be compiled without taking advantage of the built-in version of the IEEE std_logic_1164 package. This will typically result in longer simulation times for VHDL programs that use variables and signals of type std_logic.

- -noaccel <package_name>

  (optional) Turns off acceleration of the specified package in the source code using that package.

  <package_name> — A VHDL package name.

- -nocasestaticerror

  (optional) Suppresses case statement static warnings. VHDL standards require that case statement alternative choices be static at compile time. However, some expressions which

are globally static are allowed. This switch prevents the compiler from warning on such expressions. If the -pedanticerrors switch is specified, this switch is ignored.

- -nocheck

  (optional) Disables index and range checks. You can disable these individually using the -noindexcheck and -norangecheck arguments, respectively.

- -nocoverexcludedefault

  (optional) Allows VHDL code coverage data collection for the OTHERS branch in both Case statements and Selected Signal Assignment statements.

- -nocoverfec

  (optional) Disables focused expression coverage (FEC) for coverage collection. By default, FEC coverage statistics are disabled for collection. You can customize the default behavior with the CoverFEC variable in the *modelsim.ini* file.

- -nocoverrec

  (optional) Disables rapid expression coverage (REC) for coverage collection. By default, REC coverage statistics are enabled for collection. Disabling REC collection converts non-masking conditions in FEC tables to matching input patterns. You can customize the default behavior with the CoverREC variable in the *modelsim.ini* file.

- -nocoverrespecthandl

  (optional) Specifies that you want the VHDL 'H' and 'L' input values on conditions and expressions to be automatically converted to ‘1’ and ‘0’, respectively. By default in the current release, they are not automatically converted.

  As an alternative to using this argument — if you are not using 'H' and 'L' values and don’t want the additional UDP rows that are difficult to cover — you can either:

  - Change your VHDL expressions of the form (a = '1') to (to_x01(a) = '1') or to std_match(a,'1'). These functions are recognized and serve to simplify the UDP tables
  - Set the variable CoverRespectHandL in the *modelsim.ini* file to 0.

- -nocovershort

  (optional) Disables short circuiting of expressions when coverage is enabled. Short circuiting is enabled by default. You can customize the default behavior with the CoverShortCircuit variable in the *modelsim.ini* file.

- -nocoversub

  (optional) Disables code coverage data collection in VHDL subprograms. By default code coverage data is collected for VHDL subprograms. Edit the CoverSub variable in the *modelsim.ini* file to set a permanent default.

- -nocoverudp

  (optional) Prevents UDP coverage statistics from being collected. By default, only FEC coverage statistics are enabled for collection. You can customize the default behavior with the CoverUDP variable in the *modelsim.ini* file.

- -nocreatelib

  (optional) Stops automatic creation of missing work libraries and reverts back to 10.3x and earlier version behavior. Overrides the CreateLib *modelsim.ini* variable.

- -nodebug[=ports]

  (optional) Hides the internal data of all compiled design units within the GUI and other parts of the tool.

  -nodebug — The switch, specified in this form, does not hide ports, due to the fact that the port information may be required for instantiation in a parent scope.

  The design units' source code, internal structure, registers, nets, and so on, will not display in the GUI.

  In addition, none of the hidden objects may be accessed through the Dataflow or Schematic window or with commands.

  This also means that you cannot set breakpoints or single step within this code. It is advised that you not compile with this switch until you are done debugging.

  Note that this is not a speed switch like the "nodebug" option on many other products. Use the vopt command to increase simulation speed.

  -nodebug=ports — additionally hides the ports for the lower levels of your design; it should be used only to compile the lower levels of the design. If you hide the ports of the top level you will not be able to simulate the design.

  Do not use the switch in this form when the parent is part of a vopt -pdu (bbox) flow or for mixed language designs, especially for Verilog modules to be instantiated inside VHDL.

  This functionality encrypts entire files. The `protect compiler directive allows you to encrypt regions within a file.

  Design units or modules compiled with -nodebug can only instantiate design units or modules that are also compiled -nodebug.

- -nodbgsym

  (optional) Disables the generation of the symbols debugging database in the compiled library.

  The symbols debugging database is the *.dbs* file in the compiled library that provides information to the GUI allowing you to view detailed information about design objects at the source level. Two major GUI features that use this database include source window annotation and textual dataflow.

  You should only specify this switch if you know that anyone using the library will not require this information for design analysis purposes.

- -noDeferSubpgmCheck

  (optional) Causes range and length violations detected within subprograms to be reported as errors (instead of as warnings). As an alternative to using this argument, you can set the NoDeferSubpgmCheck variable in the *modelsim.ini* file to a value of 1.

- -nofprangecheck

  (optional) Disables range checks on floating type values only.

- -noFunctionInline

  (optional) Turns off VHDL subprogram inlining for design units using a local copy of a VHDL package. This may be needed in case the local package has the same name as an MTI supplied package.

- -noindexcheck

  (optional) Disables checking on indexing expressions to determine whether indexes are within declared array bounds.

- -nologo

  (optional) Disables display of the startup banner.

- -nonstddriverinit

  (optional) Forces Questa SIM to match pre-5.7c behavior in initializing drivers in a particular case. Prior to 5.7c, VHDL ports of mode out or inout could have incorrectly initialized drivers if the port did not have an explicit initialization value and the actual signal connected to the port had explicit initial values. Depending on a number of factors, Questa SIM could incorrectly use the actual signal's initial value when initializing lower level drivers. Note that the argument does not cause all lower-level drivers to use the actual signal's initial value. It does this only in the specific cases where older versions used the actual signal's initial value.

- -noothersstaticerror

  (optional) Disables warnings that result from array aggregates with multiple choices having "others" clauses that are not locally static. If -pedanticerrors is specified, this switch is ignored.

- -nopsl

  (optional) Instructs the compiler to ignore embedded PSL assertions and prevents parsing of any code within the PSL metacomment including any HDL code. By default vcom parses any PSL assertion statements it finds in the specified files. Refer to “Simulating Assertions” for more information.

- -norangecheck

  (optional) Disables run time range checking. In some designs, this results in a 2X speed increase. Range checking is enabled by default or, once disabled, can be enabled using -rangecheck. If you run a simulation with range checking disabled, any scalar values that are out of range are indicated by showing the value in the following format: ?(N) where N is the current value. Refer to “Range and Index Checking” for additional information. I

- -note <msg_number> [,<msg_number>, …]

  (optional) Changes the severity level of the specified message(s) to "note. Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

  <msg_number> — A number identifying the message. Multiple message numbers are specified as a comma separated list.

- -novital

  (optional) Causes vcom to use VHDL code for VITAL procedures rather than the accelerated and optimized timing and primitive packages built into the simulator kernel. Allows breakpoints to be set in the VITAL behavior process and permits single stepping through the VITAL procedures to debug your model. Also all of the VITAL data can be viewed in the Locals or Objects windows.

- -novitalcheck

  (optional) Disables Vital level 1 and Vital level 0 checks defined in section 4 of the Vital-95 Spec (IEEE Std 1076.4-1995).

- -novopt

  (optional) Forces vcom to produce code if the VoptFlow variable is set to 1 (optimizations turned on) in the *modelsim.ini*. (VoptFlow = 1 is the default behavior.) Use this argument together with vsim -novopt to run the simulator without any optimizations. Use of this switch is not recommended except when using Power Aware simulation together with QuestaADMS. If this switch is used, the following warning is produced:

  ```
  ** Warning: (vopt-%d) All optimizations are turned off because the
  -novopt switch is in effect. This will cause your simulation to run
  very slowly. If you are using this switch to preserve visibility for
  Debug or PLI features, please see the User's Manual section on
  Preserving Object Visibility with vopt.
  ```

- -nowarn <category_number>

  (optional) Selectively disables a category of warning messages. Warnings may be disabled for all compiles via the Main window **Compile > Compile Options** menu command or the *modelsim.ini* file (Refer to modelsim.ini Variables).

  <category_number> — Specifies one or more numbers corresponding to the categories in Table 2-11 Multiple message categories are specified as a comma separated list.

Table 2-11. Warning Message Categories for vcom -nowarn

Category number	Description
1	unbound component
2	process without a wait statement
3	null range

Table 2-11. Warning Message Categories for vcom -nowarn (cont.)

Category number	Description
4	no space in time literal
5	multiple drivers on unresolved signal
6	VITAL compliance checks (“VitalChecks” also works)
7	VITAL optimization messages
8	lint checks
9	signal value dependency at elaboration
10	VHDL-1993 constructs in VHDL-1987 code
11	PSL warnings
13	constructs that coverage can't handle
14	locally static error deferred until simulation run

- -O0 | -O1 | -O4 | -O5

  (optional) Controls optimization aggressiveness. Please refer to the section "Optimizing Designs with vopt" in the User’s Manual for detailed information on using vopt to perform optimization.

  -O0 — (capital oh zero) Enable a minimum level of optimization. Use this to work around bugs, increase your debugging visibility on a specific cell, or when you want to place breakpoints on source lines that have been optimized out. Can negatively impact performance. Useful for analyzing the behavior of the simulator.

  -O1 — Enable PE-level optimization. Note that changing from the default -O4 to -O1 may cause event order differences in your simulation. Can negatively impact performance. Useful for analyzing the behavior of the simulator.

  -O4 — (default) Enable standard SE optimizations with -O4. The main differences between -O4 and -O1 are that Questa SIM attempts to improve memory management for vectors and accelerate VITAL Level 1 modules with -O4.

  -O5 — Enable maximum optimization with -O5. Attempts to optimize loops and prevents variable assignments in situations where a variable is assigned but is not actually used. Using the +acc argument to vcom will cancel this latter optimization.

- -oldconfigvis

  (optional) Forces vcom to process visibility of VHDL component configurations consistent with prior releases. Default behavior is to comply with Language Reference Manual visibility rules. Refer to vcom -lrmconfigvis or the *modelsim.ini* variable OldVHDLConfigurationVisibility for more information.

- -optionset <optionset_name>

  (optional) Calls an optionset as defined in the *modelsim.ini file*. Refer to the section "Optionsets" for more information.

- -outf <filename>

  (optional) Specifies a file to which the final list of options is saved, after recursively expanding all -f, -file and -F files.

- -pedanticerrors

  (optional) Forces display of an error message (rather than a warning) on a variety of conditions. Refer to "Enforcing Strict 1076 Compliance" for a complete list of these conditions. This argument overrides -nocasestaticerror and -noothersstaticerror (refer above).

  You can also view a complete list of errors by executing the command:

  ```
  verror -kind vcom -pedanticerrors
  ```

- -performdefaultbinding

  (optional) Enables default binding when it has been disabled via the RequireConfigForAllDefaultBinding option in the *modelsim.ini* file.

- -preserve

  (optional) Forces vcom to preserve the case of letters in object identifiers. Can also be enabled by setting the PreserveCase variable to 1 in the *modelsim.ini* file.

- -[w]prof=<filename>

  (optional; -prof and -wprof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- +protect [=<filename>]

  (optional) Enables `protect directives for encrypting selected regions of your design source code. Produces an encrypted output file to the work library and appends the letter 'p' to the extension, for example, *vhdp*. If you specify a filename the output is the concatenation of all of the encrypted versions of the input source files.

  <filename> — A user specified string that specifies the name for the encrypted output file. Creates an encrypted output file in the current directory.

- -pslext

  (optional) Enables PSL LTL and OBE operators. These operators are disabled by default.

- -pslfile <filename>

  (optional) Compiles an external PSL assertion file together with the VHDL source files. Refer to "Simulating Assertions" for more information.

  <filename> — An existing PSL assertion file.

- -quiet

  (optional) Disables 'Loading' messages.

- -rangecheck

  (default) Enables run time range checking. Range checking can be disabled using the -norangecheck argument. Refer to "Range and Index Checking" for additional information.

- -refresh <primary> [<secondary>]

  (optional) Regenerates a library image. By default, the work library is updated. To update a different library, use -work <library_name> with -refresh (for example, vcom -work <your_lib_name> -refresh).

  <primary> [<secondary>] — Specifies the entity, package, configuration, or module to be deleted.

  This option is not supported for SystemC modules.

  - If <primary> is an entity — only that entity, no related architectures, is refreshed.
  - If <primary> is a package — the only legal value of <secondary> is "body", and only the package is refreshed.
  - If you specify both <primary> and <secondary> — Only the <secondary> architecture is updated, not the entity.

  If a dependency checking error occurs which prevents the refresh, use the vcom -force_refresh argument. Refer to the vcom Examples for more information. You may use a specific design name with -refresh to regenerate a library image for that design, but you may not use a file name.

  You cannot specify the <filename> argument when specifying this argument.

- -s

  (optional) Instructs the compiler not to load the standard package. This argument should only be used if you are compiling the standard package itself.

- -separateConfigLibrary

  Allows the declaration of a VHDL configuration to occur in a different library than the entity being configured. Strict conformance to the VHDL standard (LRM) requires that they be in the same library.

  This argument must be used if optimization is disabled (-novopt).

- -showsubprograms | -noshowsubprograms

  (optional) Toggles viewing VHDL subprogram scopes on the command line and in GUI windows, for example, the Structure window. The default is not to show subprogram scopes.

- -skip abcep

  (optional) Directs the compiler to skip all:

  a — architectures

  b — bodies

  c — configurations

  e — entities

  p — packages

  Any combination in any order can be used, but one choice is required if you use this switch.

- -skipsynthoffregion

  (optional) Ignore all constructs within synthesis_off or translate_off pragma regions.

- -smartdbgsym

  (optional) Reduces the size of design libraries by minimizing the amount of debugging symbol files generated at compile time.

  Code Coverage flows do not support this option and there are limitations to `macro support in refresh flows.

  Edit the SmartDbgSym variable in the *modelsim.ini* file to set a permanent default.

- -source

  (optional) Displays the associated line of source code before each error message that is generated during compilation. By default, only the error message is displayed.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]]

  (optional) Controls display of compiler statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the settings in the Stats *modelsim.ini* variable.

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

cmd — (default) Echo the command line.

msg — (default) Display error and warning summary at the end of command execution.

none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

perf — Display time and memory performance statistics.

time — (default) Display Start, End, and Elapsed times.

Modes

Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcom -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcom -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

kb — Print performance statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

Note

You can disable all -stats features by specifying vcom -quiet.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress. You cannot suppress Fatal or Internal messages. Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

  <msg_number> — A number identifying the message. Multiple message numbers are specified as a comma separated list.

- -togglecountlimit <int>

  (optional) Specifies a toggle coverage count limit for a toggle node. After the limit is reached, further activity on the node is ignored for toggle coverage. All possible transition edges must reach this count for the limit to take effect. For example, if you are collecting toggle data on 0->1 and 1->0 transitions, both transition counts must reach the limit. If you are collecting "full" data on 6 edge transitions, all 6 must reach the limit. Overrides the global value set by the ToggleCountLimit *modelsim.ini* variable.

  <int> — Any non-negative integer with a maximum positive value of a 32-bit signed integer and a default of 1.

- -toggleportsonly

  (optional) Enables only ports for inclusion in toggle coverage numbers; no internal signals are included in coverage metrics. If you want to see coverage of all ports in the design, use

the “vopt +acc=p” command — but note that there could be a significant performance penalty due to the fact that inlining is turned off. You can selectively enable toggle coverage on specific ports by using the “vopt +acc=p+<selection>” command. Overrides any global value set by the TogglePortsOnly *modelsim.ini* variable.

- -togglewidthlimit <int>

  (optional) Sets the maximum width of signals, <int>, that are automatically added to toggle coverage with the -cover t argument. Can be set on design unit basis. Overrides the global value of the ToggleWidthLimit *modelsim.ini* variable.

  <int> — Any non-negative integer with a maximum positive value of a 32-bit signed integer and a default of 128.

- -version

  (optional) Returns the version of the compiler as used by the licensing tools.

- -vitalmemorycheck

  (optional) Enables VITAL level 1 checks.

- -vmake

  (optional) Generates a complete record of all command line data and files accessed during the compile of a design. This data is then used by the vmake command to generate a comprehensive makefile for recompiling the design library. By default, vcom stores compile data needed for the -refresh switch and ignores compile data not needed for -refresh. The -vmake switch forces inclusion of all file dependencies and command line data accessed during a compile, whether they contribute data to the initial compile or not. Executing this switch can increase compile time in addition to increasing the accuracy of the compile. refer to the vmake command for more information.

- -vopt

  (optional) Notifies vcom that the vopt command will be run. As a result, vcom does not produce code. This saves an unnecessary code generation step. Only needed if VoptFlow is set to 0 in the *modelsim.ini*. If VoptFlow is set to 1, the vcom code generation step is skipped automatically.

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information.

  <msg_number> — A number identifying the message. Multiple message numbers are specified as a comma separated list.

- -warning error

  (optional) Reports all warnings as errors.

- -work <library_name>

  (optional) Maps a library to the logical library work. By default, the compiled design units are added to the work library. The specified pathname overrides the pathname specified for work in the project file.

  <library_name> — A logical name or pathname of a library.

- <filename>

  (required, except for when you specify -referesh or -force_refresh) Specifies the name of a file containing the VHDL source to be compiled. One filename is required; multiple filenames can be entered separated by spaces .Wildcards may be used, for example, **.vhd*.

  If you don't specify a filename, and you are using the GUI, a dialog box pops up allowing you to select the options and enter a filename.

Examples

- Compile the VHDL source code contained in the file *example.vhd*.

```
vcom example.vhd
```

- Questa SIM supports designs that use elements conforming to the 1987, 1993, and 2002 standards. Compile the design units separately using the appropriate switches.

```
vcom -87 o_units1.vhd o_units2.vhd
vcom -93 n_unit91.vhd n_unit92.vhd
```

- Hide the internal data of *example.vhd*. Models compiled with -nodebug cannot use any of the Questa SIM debugging features; any subsequent user will not be able to see into the model.

```
vcom -nodebug example.vhd
```

- The first line compiles and hides the internal data, plus the ports, of the lower-level design units, *level3.vhd* and *level2.vhd*. The second line compiles the top-level unit, *top.vhd*, without hiding the ports. It is important to compile the top level without =ports because top-level ports must be visible for simulation.

```
vcom -nodebug=ports level3.vhd level2.vhd
vcom -nodebug top.vhd
```

- When compiling source that uses the numeric_std package, this command turns off acceleration of the numeric_std package, located in the ieee library.

```
vcom -noaccel numeric_std example.vhd
```

- Although it is not obvious, the = operator is overloaded in the std_logic_1164 package. All enumeration data types in VHDL get an "implicit" definition for the = operator. So while there is no explicit = operator, there is an implicit one. This implicit declaration can be hidden by an explicit declaration of = in the same package (LRM Section 10.3). However, if another version of the = operator is declared in a different package than that

containing the enumeration declaration, and both operators become visible through use clauses, neither can be used without explicit naming.

```
vcom -explicit example.vhd
```

To eliminate that inconvenience, the VCOM command has the -explicit option that allows the explicit = operator to hide the implicit one. Allowing the explicit declaration to hide the implicit declaration is what most VHDL users expect.

```
ARITHMETIC."="(left, right)
```

- The -work option specifies mylib as the library to regenerate. -refresh rebuilds the library image without using source code, allowing models delivered as compiled libraries without source code to be rebuilt for a specific release of Questa SIM (4.6 and later only).

```
vcom -work mylib -refresh
```

- The -fsmmultitrans option enables detection and reporting of multi-state transitions when used with the +cover f argument.

```
vcom +cover=f -fsmmultitrans
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcom -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all *modelsim.ini* settings and then enables the perf option.

```
vcom -stats=time,cmd,msg -stats=none,perf
```

vcover attribute

This command is used to display attributes in the currently loaded database, during batch mode simulation, on the following types of attributes:

- Test Attributes — sets the value of attributes for testcase information. Refer to the section "Predefined Attribute Data" for complete list of these attributes.

This command provides additional information with the -help or -h switch.

Syntax

For test attributes

vcover attribute **<ucdb_file> | <coverstore>[:test]**
[-test <testname>] [-tcl] [-32 | -64] [-concise]
[-history <history_name>] [-modelsimini <path/modelsim.ini>]
[-name <attribute> ...]
[-[w]prof=<filename>] [-proftick=<integer>]
[-stats [=[+ | -]<feature>[,[+ | -]<mode>]]
[-suppress <msg_number>[,<msg_number>,…]]
[-warning <msg_number>[,<msg_number>,…]]
[-error <msg_number>[,<msg_number>,…]]
[-note <msg_number>[,<msg_number>,…]]

To display global UCDB attributes

vcover attribute -ucdb [-name <attribute> ...] **<ucdb_file> | <coverstore>[:test]**

Arguments

- -32 | -64

  (optional) Specifies whether vcover attribute uses the 32- or 64-bit executable. 32 bit is the default.These options apply only to the supported Linux operating systems.

- -concise

  (optional) Prints attribute values only, without the attribute names, for the resulting query. This is useful in some circumstances for shorter processing.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information.

- -history <history_name>

  (optional) Displays all attributes on the <history_name>. Can be used with the -name argument to specify the attribute(s) to be displayed.

- <ucdb_file> | <coverstore>[:test]

  (either UCDB or coverstore is required) The coverage database you want to analyze. You can specify a UCDB or a coverstore (created by vsim -coverstore or vcover merge -outputstore command).

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable.

  <path/modelsim.ini> — Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -name <attribute> ...

  (optional) Adds named attribute to selected test UCDBs and reports data for the specified attribute. You can specify -name <attribute> any number of times.

  <attribute> — The name of a test attribute, for example: TIMEUNIT. Multiple attributes are specified as a space separated list. Refer to Predefined Attribute Data" for a complete list of these attributes.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

time — (default) Display Start, End, and Elapsed times.

Modes

Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover attribute -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover attribute -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

kb — Print performance statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

- -note <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "note." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -[w]prof=<filename>

  (optional; -prof and -wprof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the suppress variable in the *modelsim.ini* file to set a permanent default.

- -tcl

  (optional) Prints attribute information in a Tcl list format. This allows the returned results to be stored in a Tcl list, which can be taken as input by other scripts to further process the queried data.

- -test <testname>

  (optional) Reports attribute data for the specified testname. Without this argument, all tests are reported . Wildcards are not supported. This argument is most useful when reporting on merged UCDB files that contain many tests.

  <testname> — The name of a test.

- -ucdb

  (optional) Required to display global attributes in a specified UCDB to the screen.

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

Examples

- Report all attributes of the file test.ucdb

```
vcover attribute test.ucdb
```

- Report only the USERNAME and HOSTNAME attributes for the file test.ucdb

```
vcover attribute test.ucdb -name USERNAME -name HOSTNAME
```

- Display ALL attributes on the "merge1" history node:

```
vcover attribute -history merge1
```

- Display ALL attributes on the "merge1" history node, within a specified UCDB:

```
vcover attribute -history merge1 my.ucdb
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcover attribute -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vcover attribute -stats=time,cmd,msg -stats=none,perf
```

Related Topics

- Verification Management Overview
- Verification Management Browser Window
- Verification Tracker Window
- coverage attribute
- coverage exclude
- coverage goal
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage analyze
- coverage testnames
- coverage weight
- vcover merge
- vcover ranktest
- vcover stats

vcover diff

The vcover diff command reports the coverage differences between two UCDBs. It can also write the results of the diff into a trend database (.tdb) and open it for viewing in the Results Analysis window.

Instead of showing all scopes in the design (which may be very large), you can filter the output.

If you use vcover diff with UCDBs produced by versions prior to 10.1, you may receive false positive results. You can determine whether you are getting false positives by diffing a file against itself.

The -gui switch can be used to bring up the results into the GUI. For information on viewing the results in the GUI, Refer to "Viewing vcover diff Results in the Results Analysis Window".

For information on interpreting textual output of the diff, including the BNF syntax, refer to the UCDB API Reference Manual.

This command provides additional information with the -help or -h switch.

Syntax

vcover diff **<UCDB1> <UCDB2>** <diff options>

where <diff options> are:
[-32 | -64] [-append] [-debug] [-delimiter] [-diffsfile] [-gui] [-flagfield] [-memblk] [-nocomments] [-nocounts] [-notdb]
[-[w]prof=<filename>] [-proftick=<integer>] [-rulesfile <rulesfile>] [-scope <path/scope>] [-stats [=[+ | -]<feature>[,[+ | -]<mode>]] [-structural] [-tdb <name>] [-testname <name>] <UCDB1> <UCDB2> [-verbose]

Arguments

- -32 | -64

  (optional) Specifies whether vcover diff uses the 32- or 64-bit executable. 32 bit is the default. These options apply only to the supported Linux operating systems.

- -append

  (optional) Appends differences to triage file. Default is to overwrite diff results.

- -debug

  (optional) Enables rulesfile debugging output, passed directly to the VRM transform command. Refer to the Verification Run Manager user manual for further information on the transform command.

- -delimiter

  (optional) Specifies the data delimiter. Default is TCL style string quoting, an opening and closing curly brace{}. You can change to any two characters, but the default rules file (a rules file is necessary for the .tdb creation) expects the curly braces {}. Creation of the .tdb

file, and therefore GUI viewing, will not be successful with an alternate delimiter unless you change the rules file.

If a single character is specified, it is used as both opening and closing delimiter. If two are specified, the first is used as the opening and the second as the closing.

You can specify a double quote string quoting delimiter with an entry such as:

```
vcover diff -delimiter '"' a.ucdb b.ucdb
```

- -diffsfile

  (optional) Text diff report output filename. Default is to results to stdout.

- -gui

  (optional) Open the Results Analysis window in the GUI and populate with the specified .tdb. The default file read is the QuestaDiffs.tdb.

- -flagfield

  (optional) Displays flag field as single hex number difference. Default is to display flag bits individually.

- -memblk

  (optional) Specify a limit on the number of MEMBLK bytes printed in details. Bytes are printed in hex format, in storage order from beginning of block. Does not change comparison, only display of difference.

- -nocomments

  (optional) Omits comment lines from the output.

- -nocounts

  (optional) Omits count diffs from the output.

- -notdb

  (optional) Prevents the triage database creation. Overrides -tdb, if specified.

- -[w]prof=<filename>

  (optional; -wprof and -prof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- -rulesfile <rulesfile>

  (optional) Specifies the triage rulesfile to use for creation of .tdb. Searches for the default rulesfile (*ucdbdiff.rules*) in the current working directory, and then *$MTI_HOME/vm_src/* directory. It is possible to write your own and specify that file with this command.

- -scope <path/scope>

  (optional) Filters the data to show only the specified design unit, testplan, or design hierarchical scope. <path/scope> must specify the full hierarchical path to a scope. No wildcards are allowed.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

  time — (default) Display Start, End, and Elapsed times.

  Modes

  Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover diff -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover diff -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

  kb — Print performance statistics in Kb units with no auto-scaling.

  list — Display statistics in a Tcl list format when available.

  verbose — Display verbose statistics information when available.

- -structural

  (optional) Reports only structural differences.

- -tdb <name>

  (optional) Specifies the name for the triage database created. Default base name matches that specified with -diffsfile. Otherwise, it creates a file called QuestaDiffs.tdb.

- -testname <name>

  (optional) Specifies the name for the testplan column in the GUI. The default name is QuestaDiffs.

- <UCDB1>

  (required) Designates input filename (include .ucdb file extension) to be compared against <UCDB2>.

- <UCDB2>

  (required) Designates input filename (include .ucdb file extension) to be compared against <UCDB1>.

- -verbose

  (optional) Enables increased verbosity output, useful for rulesfile debugging. Passed directly to the VRM transform command. Refer to the Verification Run Manager User's Manual for further information on the transform command.

Examples

- Reports coverage differences between covergroups in project1.ucdb and project2.ucdb into a text output, which is printed to stdout.

```
vcover diff project1.ucdb project2.ucdb
```

- Reports coverage differences between covergroups in project1.ucdb and project2.ucdb into the GUI, in the Results Analysis window.

```
vcover diff project1.ucdb project2.ucdb -gui
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcover diff -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vcover diff -stats=time,cmd,msg -stats=none,perf
```

For information on interpreting textual output of the diff, including the BNF syntax, refer to the UCDB API Reference Manual.

For information on viewing the results in the GUI, refer to the Verification Management User's Manual.

Related Topics

- Code Coverage
- Viewing vcover diff Results in the Results Analysis Window
- vcover attribute
- vcover dump
- vcover merge
- vcover ranktest
- vcover report
- vcover stats
- vcover testnames

vcover dump

This command produces and prints to *stdout* a textual description of the contents of the UCDB file. Output may be inconsistent from release to release.

This command provides additional information with the -help or -h switch.

Syntax

vcover dump **<ucdb_file>** [<ucdb_file>] [-file <dump_text_file>] [-canonical]
[-[w]prof=<filename>] [-proftick=<integer>]
[-suppress <msg_number>[,<msg_number>,…]]
[-error <msg_number>[,<msg_number>,…]]
[-note <msg_number>[,<msg_number>,…]]
[-stats [=[+ | -]<feature>[,[+ | -]<mode>]]
[-warning <msg_number>[,<msg_number>,…]]

Arguments

- <ucdb_file>

  (required) The name of the UCDB file to be printed out.

- -32 | -64

  (optional) Specifies whether vcover dump uses the 32- or 64-bit executable. 32 bit is the default. These options apply only to the supported Linux operating systems.

- -file <dump_text_file>

  (optional) Prints the text output to a specified file, instead of to *stdout*.

  <dump_text_file> — Any string representing a valid file name.

- -canonical

  (optional) Sorts nodes and tags at each level of the UCDB. Presents output in an organized, predictable manner, enabling easier comparison of multiple *.ucdb* files. It is limited to 1024 levels, and is less memory efficient than an ordinary dump.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -note <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "note." Does not function with internal messages (those without numbers).

<msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -[w]prof=<filename>

  (optional; -wprof and -prof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

  time — (default) Display Start, End, and Elapsed times.

  Modes

  Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover dump -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover dump -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

  kb — Print performance statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the suppress variable in the *modelsim.ini* file to set a permanent default.

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

Examples

- Print a textual output of the file *test.ucdb*

  vcover dump test.ucdb

- Print a canonical version of the output of file *test.ucdb*

  vcover attribute -file test.text -canonical test.ucdb

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcover dump -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vcover dump -stats=time,cmd,msg -stats=none,perf
```

Related Topics

- vcover merge
- vcover ranktest
- vcover stats

vcover history

This command allows you to acces the details on how a specific UCDB file was created, regardless of whether that UCDB was created by multiple commands over time, through scripts or regression runs, and so on.

The command functions on merged/trend UCDB files to output a list — in script form — of commands that were used to create any and all UCDB's, which were then merged to create the specified (top level) file. By default, the output includes both the xml2ucdb (testplan) and vcover merge (merge) commands required in order to re-create the merged UCDB. Options are provided to control further detail that can be output from the command.

The vcover history command's output can then be used to re-create the UCDB, if desired.

This command provides additional information with the -help or -h switch.

Syntax

vcover history < hist_options > **<file>**

```
|< hist_options > =     [-32 | -64] [-full]
                        [-display|-nodisplay ( test | merge | testplan )]*
                        [-[w]prof=<filename>] [-proftick=<integer>]
[-stats [=[+ | -]<feature>[,[+ | -]<mode>]]
```

Arguments

- <file>

  (required) The name of the Trend UCDB (.tdb), UCDB or merged UCDB (.ucdb) file to be queried.

- -32 | -64

  (optional) Specifies whether vcover history uses the 32- or 64-bit executable. 32 bit is the default. These options apply only to the supported Linux operating systems.

- -display|-nodisplay (test | merge | testplan)

  (optional) Use the -[no]display argument to prune the list of commands used to create the specified UCDB to exclude (or include) information on the nodes in the hierarchy.

  test - includes/excludes the commands used to create a leaf level or single test UCDB file. By default, this option is set to -nodisplay test. This means that the test creation commands are not included in the output.

  merge - includes/excludes the commands used to create the merged UCDB. By default this option is set to -display merge, meaning that the "vcover merge" creation commands are included in the output.

  testplan - includes/excludes the commands used to create the testplan UCDB. By default, this option is set to -display testplan, meaning that the "xml2ucdb" creation commands are included in the output.

- -full

  (optional) Prints out a full display of commands, including the complete list of commands (including “cd”) used to create the specified UCDB.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

  time — (default) Display Start, End, and Elapsed times.

  Modes

  Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover history -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover history -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

  kb — Print performance statistics in Kb units with no auto-scaling.

  list — Display statistics in a Tcl list format when available.

  verbose — Display verbose statistics information when available.

- -[w]prof=<filename>

  (optional; -prof and -wprof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

Examples

- Print a list of all commands that were used to create a simple UCDB file, *test.ucdb*:

  vcover history tracker.ucdb

  An excerpt of the resulting output would look something like this:

```
xml2ucdb -ucdbfilename <path>/VRMDATA/tracker.ucdb -format Excel -verbose
/Concat/concat_excel.xml
vcover merge -out <path>/VRMDATA/tracker.ucdb <path>/VRMDATA/tracker.ucdb
<path>/VRMDATA/regression/Randtest~2/Randtest~2.ucdb
vcover merge -out <path>/VRMDATA/tracker.ucdb /VRMDATA/tracker.ucdb
<path>/VRMDATA/regression/Randtest~3/Randtest~3.ucdb
vcover merge -out <path>/VRMDATA/tracker.ucdb
...
...
```

- Print a list of all commands that were used to create a simple UCDB file, *test.ucdb* including the command used to merge the individual leaf-leve UCDBs:

  vcover history tracker.ucdb -display test

- Print a list of commands that were used to create a Trend UCDB file, *t1.tdb* and exclude any testplan creation information:

  vcover history t1.ucdb -nodisplay testplan

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcover history -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vcover history -stats=time,cmd,msg -stats=none,perf
```

Related Topics

- vcover merge
- vcover ranktest
- vcover stats

vcover merge

The vcover merge command merges multiple code or functional coverage data files that were created with the coverage save command into a single UCDB output file. The command adjusts its functionality to the type of coverage files (functional coverage or code coverage) given on its command line.

All files being merged must have been created from the same design.

By default, vcover merge creates a basic level of information for the test. To create a test-associated merge, which associates coverage items with the test(s) that covered them, use vcover merge -testassociated. Test-associated merging is required if you plan to use:

- coverage analyze command, or any Test Analysis features in the GUI
- vcover ranktest command, or any Ranking operations in the GUI

There are cases in which it may be advisable not to merge, and instead preserve the individual UCDBs for analysis and ranking. In the following cases, the tool issues a warning message (#6846) indicating that the resulting merged file cannot completely represent the merged information:

- if "at_least" is greater than 1
- weights are different for the same object in different files
- different objects in different files

For these, you can run with the -verbose argument set to obtain further details about potential issues with the merge.

For more information, refer to "Limitations of Merge for Coverage Analysis".

If a code coverage instance in the unified coverage database (UCDB) has been changed, a warning will be generated. Warnings can be disabled with the -quiet option.

The command can be invoked within the Questa SIM GUI or at the system prompt.

This command provides additional information with the -help or -h switch.

Syntax

vcover merge <merge_options> {[-out] <outfile> <file1> [<file2> ...<filen>]}

<merge_options> =
[-32 | -64]
[-and] [-append] [-backup] [-date <yymmddhhmmss>] [-inputs <file>] [-install <path>]
[[-instance <path> [-recursive]] | [-du <du_name> [-recursive] [-du [-recursive]]...]
[-ignoredusig] [-lockingfull | -lockingtimeout | -lockingnone]
[-logfile <filename> | -l <filename>] [-master <ucdb_filename>]
[-modelsimini <path/modelsim.ini>] [-notagging]
[-outputstore <output_coverstore_dir>]
[-pathseparator <char>] [-[w]prof=<filename>] [-proftick=<integer>]

[-quiet] [-showambiguity] [-stats [=[+ | -]<feature>[,[+ | -]<mode>]] [-strip <n>] [-verbose]
[-version] [-combine | -combinemax | -combinemin | -totals | -testassociated]
[-timeout <seconds>] <file1> [<file2> ...<filen>]
[-trend [-output] <trend_ucdb_output> <ucdb_input1> ... <ucdb_inputN>]
[-error <msg_number>[,<msg_number>,…]]
[-note <msg_number>[,<msg_number>,…]]
[-suppress <msg_number>[,<msg_number>,…]]
[-warning <msg_number>[,<msg_number>,…]]

Arguments

- -32 | -64

  (optional) Specifies whether vcover merge uses the 32- or 64-bit executable. 32 bit is the default. These options apply only to the supported Linux operating systems.

- -and

  (optional) Excludes statements in the output file *only* if they are excluded in all input files. By default a statement is excluded in the output merge file if the statement is excluded in any of the input files.

  Not a valid option when merging functional coverage databases.

- -append

  (optional) Specifies that progress messages are to be appended to the current log file. By default a new log file is created each time you invoke the command.

  Not a valid option when merging functional coverage databases.

- -backup

  (optional) Creates a backup UCDB output file named "< ucdb filename >._backup" during the lock-protected execution of vcover merge.

- -combine | -combinemax | -combinemin

  (optional) Merges two or more different runs of a single test, or re-joining stripped versions of a UCDB file. The differences between the three variants of the -combine*xxx* command arguments are slight, and have to do solely with the saved counts for nodes with conflicting toggle information. In all cases, warnings result from conflicts in nodes with non-toggle data.

  -combine — for nodes with conflicting toggle information, both a minimum and maximum count is saved in UCDB. Only minimum counts are saved for conflicting non-toggle data.

  -combinemax — only maximum counts are saved for both conflicting toggle and non-toggle data.

  -combinemin — only minimum counts are saved for both conflicting toggle and non-toggle data.

  Each of the arguments is mutually exclusive with -totals and -testassociated.

- -date <yymmddhhmmss>

  (optional) Overrides the timestamp of the merged file. The <yymmddhhmmss> is the timestamp value in the form of year, day, month, hour, minute, and second.

- -du <du_name> [-recursive] [-du [-recursive]]...

  (optional) Instructs the tool to merge all instances of the specified design unit in all the input UCDB files. If one -du entry is specified, one instance of the design unit is created, at the top level, which contains all the merged data. When multiple -du arguments are specified, multiple top-level instances are created in the output UCDB, one for each -du specification.

  Instance names in the output file are generated in the following format, replacing any '/' with '_' from the library names:

  Where <du_name> is [<library name>.]<primary>[(<secondary>)] and secondary name is required only for VHDL.

  <library name>. — (optional) Specifies the library name; if none is specified, then work is assumed. Must be followed by a period (.).

  <primary> — The name of the design unit.

  <secondary> — (optional) The secondary name of the design unit, required for VHDL.

  -recursive — Instructs the tool to merge the complete design subtree, from the designated design unit down.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -ignoredusig

  (optional) Instructs the tool to ignore design unit signature checking and continue merging. This argument should not be used lightly, without first validating that the differences in code between the merges of two versions of the same file are expected and approved. Refer to "Merging and Source Code Mismatches" for further details on the use of this argument.

- -inputs <file>

  (optional) Specifies a text file containing input filenames that you want to merge.

  <file> — The name of the text file to merge.

- -instance <path> [-recursive]

  (optional) Instructs the tool to merge all occurrences of the specified instances in all the input files. It then creates an output file consisting of a single instance, containing all the merged data. Multiple -instance <path> specifications are allowed.

<path> — The path for the instance(s) specified. You can change the resulting path using the -install <path> option.

-recursive — Instructs the tool to merge the complete design subtree, from the designated instance down. If not specified, just the level specified is merged.

- -install <path>

  (optional) Adds additional hierarchy on the front end of instance and object names in the data files. This argument allows you to merge coverage results from simulations that have different hierarchies. Refer to "Merge Usage Scenarios" for more information.

  <path> — A user specified path.

- -lockingfull | -lockingtimeout | -lockingnone

  (optional) Controls the locking of the merge file.

  -lockingfull — (default) Supports breaking of locks when the creating PID is dead or when a time limit has been passed. Requires “rsh” to the machine that created the merge.

  -lockingtimeout — Supports breaking of locks when a time limit has been passed. Does not require “rsh” to creating machine. Refer to the note below for asynchronous clocks.

  -lockingnone — No locking and faster performance.

Note

The -lockingtimeout version of timeout checking is only accurate when machine clocks are synchronized.

- -logfile <filename> | -l <filename>

  (optional) Outputs progress messages to a user specified log file. By default these messages are output to *vcover.log*.

  <filename> — A user specified string.

- -master <ucdb_filename>

  (optional) Creates the output ucdb by merging a UCDB you identify as a “master” UCDB. Only one master can be specified per vcover merge command. For further usage details and rules, refer to “Merging Using a Master UCDB”.

  <ucdb_filename> — Specified master file, which should be the file which contains the data you most care about going forward. This file can be a single or a merged UCDB, either containing testplan data or not. No testplan data existing outside this master file can be merged using the -master argument.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable. .

<path/modelsim.ini> — Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -notagging

  (optional) Prevents the automatic implicit testplan tagging from being performed.

- -note <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "note." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information

- [-out] <outfile>

  (Specification of -out is optional, <outfile> is required). Sends the merged output to a file. By default, the base name of the first UCDB file listed in the command is used as the file name. When -out is specified, the output file can be specified anywhere in the command.

  <outfile> — A user specified string for the merged output file name.

- -outputstore <output_coverstore_dir>

  (optional) Specifies the output directory where the raw, merged coverage output is stored, instead of a self-contained UCDB file. This is useful when for merging the outputs of lower level merges in a hierarchical merge. When you merge other UCDBs with the coverstore, you include all the coverage data accumulated in the coverstore (from the vsim -coverstore -testname command) using the path to the directory as an input to the merge. Wildcard usage is allowed for the selection of tests to be merged.

- -pathseparator <char>

  (optional) Specifies the path separator character to be used in the resulting output UCDB file. <char> must be a non-alphanumeric character, including any of the following characters within the quotation marks: "-=~!@%^&+|./:;".

- -[w]prof=<filename>

  (optional; -wprof and -prof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- -quiet

  (optional) Disable warnings when merging databases and a changed instance is encountered.

- -showambiguity

  (optional) When used, vcover merge displays both minimum and maximum counts for any conflicting toggle data in a UCDB that results from a combined merge.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

  time — (default) Display Start, End, and Elapsed times.

  Modes

  Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover merge -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover merge -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

  kb — Print performance statistics in Kb units with no auto-scaling.

  list — Display statistics in a Tcl list format when available.

  verbose — Display verbose statistics information when available.

- -strip <n>

  (optional) Removes <n> levels of hierarchy from instance and object names in the data files. This argument allows you to merge coverage results from simulations that have different hierarchies. Refer to "Merge Usage Scenarios" for more information.

  <n> — Any positive integer.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress.

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the suppress variable in the *modelsim.ini* file to set a permanent default.

- -testassociated

  (required for ranking or test analysis purposes; optional otherwise) Merges the selected databases, including all the basic information (created with the default -totals merge) as well as the associated tests and bins. This argument is mutually exclusive with -totals and -combine.

  When tests and bins are associated, each coverage count is marked with the test that caused it to be covered. For functional coverage, this means the bin count should be greater than or equal to the at_least parameter. For code coverage and assertion data, any non-zero count for a test causes the bin to be marked with the test. While it cannot be known which test incremented a bin by exactly how much, it can be known which test caused a bin to be covered.

- -totals

  (optional) Merges the databases with a basic level of information, including: coverage scopes, design scopes, and testplan scopes. This is the default merge. The counts are incremented together. In the case of vector bin counts, counts are ORed. The final output database is a union of objects from the input files. Information about which test contributed what coverage into the merge is lost. Information about tests themselves are not lost — test data records are added together from all merge inputs. While the list of tests can be known, it cannot be known what tests might have incremented particular bins. Mutually exclusive with -combine and -testassociated.

- -timeout <seconds>

  (optional) Sets the timeout period after which the lock can be removed. During the timeout period the lock holder is protected. Supports cumulative merges and multiple merge commands, issued one after another. Such merge commands can be issued simultaneously from various platforms in a networking environment. In order to avoid corrupting cumulative coverage results, merges of UCDB files are serialized.

  <seconds> — Any non-negative integer.

- -trend [-output] <trend_ucdb_output> <ucdb_input1> ... <ucdb_inputN>

  Creates a trending database used to look at coverage over time for a particular design under test (DUT). The following data is automatically added to the trend UCDB, through the addition of global attributes:

  Total number of tests

  Number of passed tests

Number of warning tests

Number of error tests

Number of fatal error tests

Total number of bins linked with testplan sections

Number of covered bins linked with testplan sections

-output — (optional) Names the output file for trend ucdb. Only required if output UCDB file is positioned other than first listed UCDB.

<trend_ucdb_output> — (required) Name of file created with trend command. Must be placed before input UCDBs, unless -output is used.

<ucdb_input> — (required) Name(s) of input UCDB file(s) to be processed for trend analysis. Files need not be merged files, but could be individual UCDB files and/or trend UCDB files.

- -verbose

  (optional) Enables summary code coverage statistics to be computed and directed to the log file each time a file is merged into the base. The statistics are instance-based.

  Not a valid option when merging functional coverage databases.

- -version

  (optional) Returns the version number of each input UCDB file, and the version number of the output UCDB file, which is always created with the most recent version of the UCDB creation software.

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- <file1> [<file2> ...<filen>]

  (required) Specifies the file(s) to be merged. Files may be in the form of either a UCDB or <coverstore>[:test]. Multiple pathnames and wildcards are allowed. The merge performed is a union of the content of all files. Refer to the Examples section.

Examples

- Merge coverage statistics for *myfile1* and *myfile2* and writes them to *myresult*.

  ```
  vcover merge myfile1 myfile2 -out myresult
  ```

- Use wildcards to merge all files with a *.cov* extension in a particular directory.

```
vcover merge myresult2 /dut/*.cov
```

- Create a trend database to be used with vcover report for trending analysis, saving the file to *mytrend.ucdb*.

```
vcover merge -trend -output mytrend.ucdb Nov_mg.ucdb Dec_mg.ucdb
```

  The *Nov_mg.ucdb* and *Dec_mg.ucdb* are two merged files from simulations of the same DUT.

- Merge two design units within a single input file to create two top level instances in the merged output.

```
vcover merge out.ucdb -du counter -du work.mux test.ucdb
```

- Change the default time-out for the lock file to 600 seconds.

```
vcover merge -timeout 600 out1.ucdb out2.ucdb in.ucdb
```

- Strip the top two levels of hierarchy from an instance or objects in *myfile.ucdb* and place into another file *myfile_stripped.ucdb*.

```
vcover merge -strip 2 myfile_stripped.ucdb myfie.ucdb
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcover merge -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vcover merge -stats=time,cmd,msg -stats=none,perf
```

Related Topics

- “Code Coverage”
- “Verification with Functional Coverage”
- “Verification with Assertions and Cover Directives”
- "Merge Usage Scenarios",
- "Verification Management Overview”
- "Merging Verification Plan with Test Data"
- “Analyzing Trends”
- coverage analyze
- coverage attribute
- coverage ranktest
- coverage save
- coverage tag
- coverage testnames
- coverage unlinked
- vcover attribute
- vcover merge
- vcover ranktest
- vcover stats

vcover parallelmerge

The vcover parallelmerge command performs a merge on a UCDB or CoverStore database using parallel processes. It automatically merges any intermediate merge results, and produces final merged results.

The command can be invoked within the Questa SIM GUI or at the system prompt.

This command provides additional information with the -help or -h switch.

Syntax

vcover parallelmerge <options>

<options> =
[-covmode <ucdb/covstore>] [-filelist <filepath>] [-genlist] [-genlistfrom <pathdir_or_list>] [-gridopts <options>] [-gridslots <val>] [-hostlist <pattern>] [-j <val>] [-mergetype <totals/testassociated/preservetestcounts>] [-mergeopts <options>] [-mtimeout <val>] [-outname <name>] [-qtimeout <val>] [-rmdb <path>] [-runmode <local/rsh/lsf/sge>][-teststatus <all/passed/failed>] [-dbname <file_pattern>] [-verbose]

Arguments

- -covmode <ucdb/covstore>

  (optional) Specifies type of coverage database, UCDB or CoverStore (Default: ucdb).

- -filelist <filepath>

  (optional) Specifies file list name and path , which contains a list of files to be merged. If list is generated by the tool, then list is generated in file name specified by this option (Default: $cwd/parallellist).

- -genlist

  (optional) Generates the file list from $cwd for UCDB mode, or $cwd/CoverStore for Coverstore mode. The file list is generated at the location specified by -filelist option. The command recursively searches the given db path with -ucdbname file name pattern and generate list of files to be merged.

- -genlistfrom <pathdir_or_list>

  (optional) Generates the file list from specified database location(s). File list is generated at the location specified by the -filelist option. The command recursively searches the specified database path — which is the -ucdbname file name pattern for UCDB mode, or <pathdir_or_list> for Coverstore mode — and generates the list of files to be merged.

  If a list of databases is entered, the list must be contained within double quotes ("").

- -gridopts <options>

  Specify grid options that can be used while launching grid jobs (Default: "")

- -gridslots <val>

  (optional) Specify number of grid slots to use (Default: 10). This sets the maximum number of parallel merge jobs on the selected grid type. -j option can be used in conjunction with -gridslots. -j will determine how many files each process will merge and -gridslots will determine how many parallel process will run on the grid at a time.

- -hostlist <pattern>

  (optional) Specifies list of hosts or file containing host list (Default: ""). Number of host provided in this list also determines maximum number of parallel jobs to run. user can also specify file containing list of hosts. Format is "host1[:<n1>][, host2[:]<n2>] ...]". File can contain multiple lines with same syntax. vcover will read this file and expand the hostlist in space delimited format. So, "host1:2, host2:3" will become "host1 host1 host2 host2 host2" and number of maximum jobs will be 5.

- -j <val>

  (optional) Specifies number of parallel merge jobs (Default: 10). These are number of maximum parallel jobs to run. The maximum number of parallel jobs is used to split the filelist into number of files merged by each parallel process. For 'rsh' -runmode type, this value is calculated from -hostlist.

- -mergetype <totals/testassociated/preservetestcounts>

  (optional) Specifies merge type the type of merge algorithm used for the merge processes (Default: totals).

- -mergeopts <options>

  (optional) Specifies other merge options (Default: "").

- -mtimeout <val>

  (optional) Specifies a merge timeout value that is used for time-outs, which is measured between the time that a merge process starts and the time the process finishes. The value (<val>) is the specific maximum runtime value from all the first or second level parallel merge processes. The runtime for each parallel merge process depends on the size of single database file, the number of total files to merge, and the number of parallel processes used. (Default: 10000).

- -outname <name>

  (optional) Specifies merged output file name prefix (Default: merge). vcover will automatically append suffix name based on -covmode.

- -qtimeout <val>

  (optional) Specify queue timeout value. This value is used for time-outs and is measured between the time that a merge process is launched to the time it actual starts to execute. (Default: 5000).

- -rmdb <path>

  (optional) Specifies user-defined RMDB file. The vcover parallelmerge command auto-generates the RMDB file in $cwd/TEMP_MERGE_DIR based on vcover options provided.

If user wants to use customized RMDB file, then they can copy auto-generated RMDB file, modify it as per their requirements and use it with this option.

- -runmode <local/rsh/lsf/sge>

  (optional) Specifies type of parallel merge run mode. Options are local, (Default: local).

- -teststatus <all/passed/failed>

  (optional) Merges tests only with specified status in TESTSTATUS value in UCDB file (Default: all). When passed or failed teststatus is specified, vcover will scan the provided or generated filelist and use only those files with given teststatus to merge.

- -dbname <file_pattern>

  (optional) Specifies UCDB or CoverStore file patterns to search in database location specified with either -genlist or -genlistfrom. (Default: *.ucdb).

- -verbose

  (optional) Prints additional verbose information.

Examples

- Specify cs1 and cs2 (Coverstore) databases from the default location for merging in parallel:

  ```
  vcover parallelmerge -genlist "cs1 cs2"
  ```

- Specify UCDB databases from the default location for merging in parallel:

  ```
  vcover parallelmerge -genlist -ucdbname "ucdbdir1 ucdbdir2"
  ```

- Specify a list of databases (cs1 and cs2) to be merged (Coverstore mode):

  ```
  vcover parallelmerge -genlistfrom "cs1 cs2"
  ```

- Specify a databases in the directory dir1 (UCDB mode):

  ```
  vcover parallelmerge -genlistfrom dir1
  ```

Related Topics

- vcover merge
- “Parallel Merge”

vcover ranktest

The vcover ranktest command ranks coverage data contained in the specified tests, according to their contribution to cumulative coverage. The tests to be ranked are taken from the UCDB files specified as inputs. These can be merged or unmerged UCDBs. If merged, they must have been created with the vcover merge -testassociated argument.

The output of this command is written to stdout (screen) and two lists of tests (contributing *ranktest.contrib* and non-contributing tests in *ranktest.noncontrib*). Any missing tests are also listed in the output. The order of the tests written to the screen or a specified file (with -log) are listed in the order shown in Table 2-12.

Table 2-12. Order and Type of Ranked Tests

Contributing, compulsory	Mandatory tests, tests which need to be run regardless of achieved coverage.	Sorted by total coverage %
Contributing, noncompulsory	Tests providing coverage not provided by any previous test.	Sorted by total coverage %
Non-contributing	Redundant tests, providing no incremental coverage.	Not sorted

To rank on a specific coverage item, design unit, or testplan section within the hierarchy of your design, use -path, -du or -plansection.

This command provides additional information with the -help or -h switch.

Syntax

vcover ranktest [<ranktest_options>] {<file1> [... <fileN>]}

<ranktest_options> =
[-32 | -64] [-algorithm {greedy | quick}]
[-assertion] [-code {b | c | e | f | s | t}...] [-codeAll] [-cvg] [-directive]
[-nocompulsoryordering | -compulsorysorting | -compulsoryranking]
[-error <msg_number>[,<msg_number>,…]]
[-fewest | -cputime | -simtime [-goal [<coverage_type>] <percentage>]]
[-inputs <file_list>] [-j <number_of_parallel_processes>]
[-logfile <filename> | -l <filename>]
[-maxcpu <real_num_in_seconds>] [-maxtests <int>]
[-metric {aggregate | total}]
[-note <msg_number>[,<msg_number>,…]]
[-path <path> | -du <du_name> | -plansection <path>] [-precision <int_num>]
[-[w]prof=<filename>] [-proftick=<integer>] [-quiet | -concise | -verbose]
[-rankfile <filename>]
[-stats [=[+ | -]<feature>[,[+ | -]<mode>]]
[-suppress <msg_number>[,<msg_number>,…]]
[-warning <msg_number>[,<msg_number>,…]]

[-weight <coverage_type> <integer>]
[-iterative | -testassociated] [-keepmergefile <filepath>] [-modelsimini <path/modelsim.ini>]

Arguments

- -32 | -64

  (optional) Specifies whether vcover ranktest uses the 32- or 64-bit executable. 32 bit is the default. These options apply only to the supported Linux operating systems.

- -algorithm {greedy | quick}

  Specifies algorithm being used to produce the ranked list of tests: greedy is the default algorithm used. It produces the shortest list of tests contributing to coverage, arranged in order of highest incremental contribution to least. The quick algorithm provides an estimated ordering of tests, not guaranteed to be the shortest list or that the order of tests is perfectly determined as to a particular test's significance.

- -assertion

  Specifies ranktest for assertion data. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

  Traditionally, assertions are a measure of correctness and not a measure of coverage. Because of this, assertion metrics are ignored in ranktest calculations when there are assertion failures. When viewed as a coverage item, their definitions are as follows:

  When an assertion has both pass and fail counts:

  it is passed if pass count > 0 and fail count == 0

  Otherwise, when it has only a fail count:

  it is passed if fail count == 0

- -code {b | c | e | f | s | t}...

  (optional) Specifies ranktest for corresponding code coverage type only: branch, condition, expression, FSM, statement, toggle. More than one coverage type may be specified with each -code argument (example: "-code bcest"). This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

- -codeAll

  (optional) Specifies ranktest for all coverage types. Equivalent to -code bcestf. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

- -nocompulsoryordering | -compulsorysorting | -compulsoryranking

  Specifies how the command ranks compulsory tests, when present.

  -nocompulsoryordering — (default) Ranks compulsory tests in a random order with no ranking with respect to each other. Does not display compulsory tests in the detailed

ranktest listing. In the GUI, the accumulated coverage data and incremental coverage data for compulsory tests is displayed as a minus sign (-). This option has the best performance of the three options.

-compulsorysorting — Ranks compulsory tests in the order of descending 'total coverage' with respect to each other. Displays compulsory tests in the detailed ranktest listing according to this order. In the GUI, the accumulated coverage data and incremental coverage data for compulsory tests is calculated and displayed. This option has performance second to the -nocompulsoryodering option.

-compulsoryranking — Ranks compulsory tests fully with respect to each other. Displays compulsory tests in the detailed ranktest listing according to this order. In the GUI, the accumulated coverage data and incremental coverage data for compulsory tests will be calculated and displayed. This option can have the worst performance of the three options.

- -concise

  (optional) Specifies the output is created with minimum additional I/O. Default creates ranktest with full I/O (-verbose). Mutually exclusive with -quiet and -verbose.

- -cputime

  (optional) Specifies that the order of tests being selected for ranking is: the largest coverage gain per unit of CPU time. Mutually exclusive with the -fewest and -simtime arguments.

- -cvg

  Specifies ranktest for covergroup data. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

- -directive

  Specifies ranktest for directive data. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

- -du <du_name>

  (optional) Restricts the ranking to the specified design unit. This argument applies to a particular module type, by name, in all UCDB files. The -du ranked coverage is the combined "local" coverage of a flat list of all instances of that DU.

  This option is mutually exclusive with -path and -plansection.

  Where <du_name> is [<library name>.]<primary>[(<secondary>)] and secondary name is required only for VHDL.

  <library name>. — (optional) Specifies the library name; if none is specified, then work is assumed. Must be followed by a period (.).

  <primary> — The name of the design unit.

  <secondary> — (optional) The secondary name of the design unit, required for VHDL.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Does not function with internal messages (those without numbers).

<msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -fewest

  (optional) Specifies that the tests be selected for ranking by: the largest coverage gain with the fewest number of tests. Mutually exclusive to the -cputime and -simtime arguments. Default.

- -inputs <file_list>

  (optional) Specifies a file containing ranktest arguments. The file can contain a list of UCDB files to be ranked.

  <file_list> — The name of the text file.

- -iterative

  (optional) Ranks the coverage items in the specified database(s) with a basic level of information, including: coverage scopes, design scopes, and testplan scopes. Mutually exclusive with -testassociated.

- -j <number_of_parallel_processes>

  (optional) Specifies the maximum number of parallel processes used during the ranktest operation.

- -goal [<coverage_type>] <percentage>]

  (optional) Specifies that the tests be ranked to achieve this coverage goal. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

  <coverage_type> — specifies that the goal is applied only to the specified type of coverage; otherwise, the goal is applied to all types of coverage.

  Valid values for <coverage_type> are: -assertion, -assertfailure, -cvg, -directive, -code {b | c | e | f | s | t}... or -codeAll.

  <percentage> — an integer value; the default is 100.

  The -goal argument can be specified multiple times, as shown in this example:

  ```
  vcover ranktest -goal -assertion 90 -goal -code bcest 95
  ```

- -keepmergefile <filepath>

  (optional) Specifies the merge file corresponding to the ranking be preserved. By default, the merge file is deleted.

  <filepath> — Specifies either an absolute or relative path to the merge file. On Windows systems the path separator should be a forward slash (/).

- -logfile <filename> | -l <filename>

  (optional) Specifies the file for outputting ranked results. Output includes full path to tests.

<filename> — A user specified string.

- -maxcpu <real_num_in_seconds>

  (optional) Monitors the accumulated CPU time of the ranked tests. Specifies the maximum CPU time to be allowed. If the specified number of seconds is exceeded, the ranking process is stopped.

  <real_num_in_seconds> — Any integer greater than -1.0 where the default is -1.0 (no limit).

- -maxtests <int>

  (optional) Specifies threshold for the maximum number of tests to be ranked. When this threshold is exceeded, the ranking operation is terminated.

  <int> — Any positive integer.

- -metric {aggregate | total}

  (optional) Indicates the kind of metric used for ranking.

  total —

  (Default, unless any of the following arguments are used: -goal, -weight, -assertion, -assertfailures, -cvg, -directive, -codeAll, or -code)

  Ranking metric used produces values consistent with the totals for the UCDB obtained with coverage analyze, which appear in Total Coverage numbers. Refer to "Calculation of Total Coverage" for details.

  aggregate — Ranking metric used produces aggregate values based on each individual coverage type: values are not likely to be consistent with totals produced with any other means. Refer to “Important” note below.

  Each coverage type can be selected or not. Each coverage type can be given an individual weight and goal. These individual numbers are then combined and normalized to yield an aggregate metric that is unrelated to the number given by the total coverage.

  When the -metric aggregate argument is used, the resulting metric number will not “match” any other total coverage number produced by other verification tools (for example, coverage analyze).

  This is important because when you use any of the arguments (-totals, -goal, -weight, -assertion, -assertfailures, -cvg, -directive, -codeAll, or -code) with ranktest command, the aggregate metric is the default.

  Refer to “Merged Results vs. Rank Report Coverage Why They Can Differ” for further details.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable.

<path/modelsim.ini> — Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -note <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "note." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information

- -path <path>

  (optional) Restricts ranking to design paths (non-testplan) matching the specified <path>. This coverage is a hierarchical coverage — a "rolling up" of the coverage — including all children of the specified path. This option is mutually exclusive with -du and -plansection.

  <path> — Specifies either an absolute or relative path to the design files. On Windows systems the path separator should be a forward slash (/).

- -plansection <path>

  (optional) Restricts ranking to the specified testplan node. This argument applies to a particular module type, by name, in all UCDB files. This option is mutually exclusive with -du and -path.

  <path> — Specifies either an absolute or relative path to the testplan. On Windows systems the path separator should be a forward slash (/).

- -precision <int_num>

  (optional) Specifies the decimal point precision for output only: The contents of the rank file are NOT affected by this argument.

  <int_num> — Any positive integer where the default is 2.

- -[w]prof=<filename>

  (optional; -wprof and -prof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- -quiet

  (optional) Creates the ranktest output without any additional I/O. Default creates ranktest with full I/O (-verbose). Mutually exclusive with -concise and -verbose.

- -rankfile <filename>

  (optional) Specifies the name the ranktest file being created. Default if not specified is *ranktest.rank*. Can be specified with the vcover stats command to redisplay the results of this ranking. This file can also be used to repopulate the Browser with ranktest information.

  <filename> — A user specified string, which can include a path.

- -simtime

  (optional) Specifies that the order of tests being selected for ranking is: the largest coverage gain per unit of simulation time. Mutually exclusive to the -cputime and -fewest arguments.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

  time — (default) Display Start, End, and Elapsed times.

  Modes

  Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover ranktest -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover ranktest -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

  kb — Print performance statistics in Kb units with no auto-scaling.

  list — Display statistics in a Tcl list format when available.

  verbose — Display verbose statistics information when available.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the suppress variable in the *modelsim.ini* file to set a permanent default.

- -testassociated

  (optional) Ranks the coverage items in the selected database(s) including all the basic information (as created with -iterative) as well as the associated tests and bins. This is the default ranktest. This argument is mutually exclusive with -iterative.

- <file1> [... <fileN>]

  (required unless the -inputs file is used) May be in the form of either a UCDB or coverstore[:test] file. Specifies the name of two or more non-merged UCDB file(s) to rank.

- -verbose

  (optional) Specifies the output is created with full I/O. Mutually exclusive with -quiet and -concise. Default

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -weight <coverage_type> <integer>

  (optional) Used when selecting next ranking candidate. This argument is not compatible with -metric total: if both are explicitly set, ranking is halted and an error is issued.

  <coverage_type> — specifies that the goal is applied only to the specified type of coverage; otherwise, the goal is applied to all types of coverage. Valid values for <coverage_type> are:

  -assertion, -assertfailure, -cvg, -directive, -code {b | c | e | f | s | t}... or -codeAll.

  <integer> — Any non-negative integer.

  The -weight argument can be specified multiple times, as shown in this example:

```
vcover ranktest -weight -assertion 10 -weight -code bcest 5
```

Examples

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcover ranktest -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vcover ranktest -stats=time,cmd,msg -stats=none,perf
```

Related Topics

- vcover merge
- vcover stats
- "Code Coverage"
- coverage goal
- coverage weight
- coverage ranktest

vcover remove

This command allows you to create a copy of a previously merged testplan UCDB, excluding the testplan section and all associated links and tags. The newly saved UCDB has no testplan data, links, or tags.

This command provides additional information with the -help or -h switch.

Syntax

vcover remove -tplan {[-out] <outfile>} <infile> [-[wprof=<filename>]
[-proftick=<integer>] [-stats [=[+ | -]<feature>[,[+ | -]<mode>]]

The <outfile> and <infile> are order dependent, unless -out is used.

Arguments

- [-out] <outfile>

  (<outfile> is required) The name of the newly created UCDB, which is to be created without the testplan section. When -out is used, the <outfile> could be specified after the <infile>.

- <infile>

  (required) The name of the original UCDB file with the testplan section to be removed.

- -[w]prof=<filename>

  (optional; -wprof and -prof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

msg — (default) Display error and warning summary at the end of command execution.

none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

perf — Display time and memory performance statistics.

time — (default) Display Start, End, and Elapsed times.

Modes

Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover remove -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover remove -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

kb — Print performance statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

Examples

- Remove the testplan, including all associated links and tags, from merged testplan UCDB, *test22.ucdb*. Save new file as *notpTest22.ucdb*:

```
vcover remove -tplan notpTest22.ucdb test22.ucdb
```

- Remove the testplan from *test22.ucdb* and save into the location */noTP/noTest22.ucdb*.

```
vcover report -tplan test22.ucdb -out /noTP/noTest22.ucdb
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcover remove -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vcover remove -stats=time,cmd,msg -stats=none,perf
```

Related Topics

- vcover merge

vcover report

The vcover report command prints textual output of coverage statistics or exclusions — from a previously saved code or functional coverage run — to a specified file.

This allows you to produce reports "offline" (for example, without having to load a simulation.)

You can choose from a number of report output options using the arguments listed below.

> **Tip**: HTML reports are created with the -html option, which is only compatible with the options listed therein. Refer to the -html section for details.

By default, the command prints out results from the current scope.

To specify a certain path for the report, you can use the -instance argument, or specify the specific cover directive or covergroup, such as:

- vcover report -instance <path>
- vcover report <path_to_cover_directive/covergroup>

The command orders the output on a by-file basis unless you specify the -byinstance or -bydu argument.

The report displays code coverage data from generate blocks.

This command provides additional information with the -help or -h switch.

Syntax

vcover report [<coverage_arguments>] **<input_ucdb>**

vcover report [-version] **<input_ucdb>**

where <coverage_arguments> can be categorized into the following groups:

Generic Coverage Arguments

vcover report <rpt_args> <ucdb_or_coverstore_test>

<rpt_args> =
[-32 | -64] [-above <percent> | -below <percent>] [-append]
[-bydu] [-byfile] [-byinstance] [-binrhs] [-comment | -nocomment]
[-details [-dumptables] [-fecanalysis] [-metricanalysis]] [-memory]
[-modelsimini <path/modelsim.ini>] [-zeros] [-nomissing] [-nocomment | -comment]
[-totals] [-nozeroweights]
[-precision <int>] [-[w]prof=<filename>] [-proftick=<integer>] [-recursive [-depth <n>]]
[-flat [-primarykey <type | path | value> [-secondarykey <type | path | value>]]]
[-showambiguity] [-stats [=[+ | -]<feature>[,[+ | -]<mode>]]
[-testextract <test_name_or_pattern>]
[-file <filename>] [-xml]
[-suppress <msg_number>[,<msg_number>,…]]

[-error <msg_number>[,<msg_number>,…]]
[-note <msg_number>[,<msg_number>,…]]
[-warning <msg_number>[,<msg_number>,…]]

<ucdb_or_coverstore_tests> =
<ucdb_file> | <coverstore>[:test]

Arguments for creating HTML output from a UCDB

vcover report -html <input_ucdb> [-code [bcesf[t|x]] [-assert] [-binrhs] [-cvg] [-directive] [-details[=abcdefgst]] [-du <path/du] ([-dynamic [-servermode |-stopserver] -port <port_no>]]) | [-htmldir <outdir>] [-hidecvginsts | -hidecvginstspi0] [-instance <path/instance>] [-nodu] [-nofec] [-noudp] [-noframes] [-noinstance] [-notestplan] [-notimestamp] [-nozeroweights] [-showexcluded] [-source] [-stmtaltflow] [-summary] [-testhitdata] [-testlist] [-threshL <val>] [-threshH <val>] [-usecnpm] [-verbose]

Filtering Arguments - Used to filter one or more coverage types in the report

vcover report [-assert] [-code {b | c | e | f | s | t}...] [-concurrent | -immediate] [-covered] [-cvg] [-directive] [-failed] [-filter <pattern>] [-lang sva | psl | vhdl] [-severity info | note | warning | error | failure | fatal] [-testattr] [-unattemptedimmed]

For each directive instance, the report includes by default the full instance path, the coverage count or percentage, the design unit, the source file name, and the source line number.

Code Coverage Arguments

Has no effect on assertion or functional coverage

vcover report [-coverenhanced] [-noannotate] [-nofec] [-noudp] [-library <libname>] [-du <du_name>] [-package <pkgname>] [-source <filename>] [-instance <path> [-recursive [-depth <n>]]

Exclusion Coverage Arguments

vcover report [-excluded [-pragma | -user] [-nocomment]] [-noexcludedhits] [-code {b | c | e | f | s | t}...] [-instance <path>] [-file <filename> [-append]]

Toggle-specific Coverage Arguments

vcover report [-toggles] [-duplicates] [-verbose] [-all] [-top] [-portmode {input | output | inout | internal}] [-select {inputs | outputs | inout | ports | internals}]

Toggle coverage statistics are relevant only when reporting on instances or design units and are not produced on a per file basis. Toggle data is summed for all instances, and is reported by port or local name in the design unit, rather than by the connected signal. If you want toggle coverage statistics, you must specify either the -byinstance, -bydu, -instance <path>, or -du <du_name> arguments. If you do not use those arguments, or you use the -source <filename> argument, toggle coverage statistics are excluded even if you specify -code t. To get an itemized list of the signals, the -details argument is also required.

To report extended toggle coverage, ensure that you have compiled (vlog/vcom) with the -code x argument, then use vcover report with -code t.

For a Report of Non-collected Coverage Items

coverage report -nocollect [-file <filename> [-append]]

Covergroup-specific Arguments

vcover report [-binrhs] [-cvg] [-hidecvginsts | -hidecvginstspi0] [-nocrossbinsummary] [-nocvpbinsummary] [-noignorebin] [-option] [-samples <spec>] [-usecnpm]

Trend Report Arguments

HTML formatted:

vcover report -trend -html <trend_ucdb> [-htmldir <outdir>] [-assert] [-cvg] [-code {b|c|e|s|t...}] [-directive] [-descendants] [-verbose] [-noframes] [-above <%>] [-below <%>] [-precision <int>] [-summary] [-linegraph] (([-from <YYYYMMDDHHMMSS | first>] | [-to <YYYYMMDDHHMMSS | last>]) [-numsamples <int>])

XML formatted:

vcover report -trend -xml <trend_ucdb> {-bydu | -byinstance | -byplan} [-file <xml_output>] [-assert] [-cvg] [-code {b|c|e|s|t...}] [-descendants] [-directive] [-attribute <name> <value> | -byattribute] [-above <%>] [-below <%>] [-precision <int>] (([-from <YYYYMMDDHHMMSS | first>] | [-to <YYYYMMDDHHMMSS | last>]) [-numsamples <int>])

Comma Separated Value (CSV) formatted —

vcover report -trend -csv <trend_ucdb> {-bydu | -byinstance | -byplan} [-file <csv_output>] [-assert] [-cvg] [-code {b|c|e|s|t...}] [-directive] [-descendants] [-attribute <attr>| -byattribute] [-above <%>] [-below <%>] [-precision <int>] (([-from <YYYYMMDDHHMMSS | first>] | [-to <YYYYMMDDHHMMSS | last>]) [-numsamples <int>])

Text formatted:

vcover report -trend <trend_ucdb> {-bydu | -byinstance | -byplan} [-du <duname>] [[-instance | -path] <path>] [-plansection <path>] [-recursive [-depth <n>] [-file <output>] [-assert] [-cvg] [-code {b|c|e|s|t|f}] [-directive] [-descendants] [-above <%>] [-below <%>] (([-from <YYYYMMDDHHMMSS | first>] | [-to <YYYYMMDDHHMMSS | last>]) [-numsamples <int>])

Plots a 2-dimensional graph, in either HTML, XML, Text format, of a trending database, used for the purpose of analyzing coverage over time for a particular design under test (DUT).

Arguments

- <input_ucdb>

  (required) Specifies the previously saved code coverage file on which you want to report.

  (required) Specifies the previously saved code or functional coverage file on which you want to report.

- -32 | -64

  (optional) Specifies whether vcover report uses the 32- or 64-bit executable. 32 bit is the default. These options apply only to the supported Linux operating systems.

- -above <percent>

  (optional) Only objects with coverage values above the specified percentage will be included in the output. Includes coverage of coverpoints and crosses, not covergroups. Refer also to -below.

  <percent> — An integer between 0 and 100.

- -all

  (optional) When reporting toggles, creates a report that lists both toggled and untoggled signals. Counts of all enumeration values are reported. Not a valid option when reporting on a functional coverage database.

- -append

  (optional) Appends the report data to the named output file.

- -assert

  (optional) Reports only assertion coverage data.

- -below <percent>

  (optional) Only objects with coverage values below the specified percentage will be included in the output. Includes coverage of coverpoints and crosses, not covergroups. Refer also to -above.

  <percent> — An integer between 0 and 100.

- -binrhs

  (optional) Specifies that for covergroups, a column is included in the report that displays the RHS for covergroup bins. The RHS is a sampled value that causes the bin to increment.

- -bydu

  (optional) Reports coverage statistics by design unit (du). The simulator will iterate through all design units/modules in the design and report coverage data for each. Each design unit report will be the merged result of all instances of that module and will be sorted by design unit name. You can also report coverage data for a specific design unit by using the -du <name> argument.

- -byfile

  (optional) Writes out a coverage summary for each source file in the design. This is the default report generated.

- -byinstance

  (optional) Writes out a coverage summary for all instances and packages. The default setting, if not used, is -byfile. Can be used with the -recursive [-depth <n>] argument to report on all instances recursively.

- -code {b | c | e | f | s | t}...

  (optional) Specifies which code coverage statistics to include in the report. If this argument is not specified, the report includes statistics for all categories you enabled at compile time.

  Not a valid option for functional coverage data.

  The characters are as follows:

  b — Include branch statistics.

  c — Include condition statistics.

  e — Include expression statistics.

  f — Include finite state machine statistics.

  s — Include statement statistics.

  t — Include toggle statistics.

  To report extended toggle coverage, ensure that you have compiled (vlog/vcom) with the -code x argument, then use vcover report with -code t.

- -comment | -nocomment

  (optional) Include or exclude comments from the report. By default, comments are included.

- -concurrent | -immediate

  (optional) Only concurrent assertions are selected when the -concurrent switch is used. Only immediate assertions are selected when the -immediate switch is used. These are mutually exclusive options. If neither switch is specified, both concurrent and immediate assertions are selected.

- -covered

  (optional) Specifies that only the covered bins of a coverpoint or cross appear in the report. This argument applies to covergroups only. If any covergroup item survives the covergroup filters applied, its ancestors are displayed in the report and the GUI.

- -coverenhanced

  (optional) Enables non-critical functionality which might change the appearance or content of coverage metrics. This argument only has an effect in letter releases (10.0a, 10.0b, and so on). In major releases (10.0, 10.1, and so on), all coverage enhancements present in previous letter release streams are enabled by default, and -coverenhanced is no longer necessary to enable these enhancements. Bug fixes important to the correctness of coverage numbers are always enabled by default, with no need for -coverenhanced. Since the exact nature of -coverenhanced varies from release to release, the details of the enhancements it enables are present in the product release notes rather than in the Command Reference. For these details, search the release notes using the string "coverenhanced".

- -cvg

  (optional) Reports only covergroup coverage data.

- -details [-dumptables] [-fecanalysis] [-metricanalysis]

  (optional) Includes details associated with each coverage item in the output for FEC coverage (and UDP, if enabled). By default, details are not provided.

  -dumptables — (optional) forces printing of condition and expression truth tables even though fully covered.

  -fecanalysis — (optional) reports which input patterns can be applied to the inputs to increment the expression/condition hit counts.

  -metricanalysis — (optional) If UDP coverage is enabled, it prints sum-of-product and basic sub-condition heuristic metrics from UDP expression/condition view. It reports hit counts for all rows in UPD table. To improve coverage numbers, find rows with 0 hits and exercise the inputs accordingly. Refer to "Condition and Expression Coverage" for more information on metrics.

- -directive

  (optional) Reports only directives coverage data.

- -du <du_name>

  (optional) Reports coverage statistics for the specified design unit. If there are parameterized instances, all are considered to match the specified design unit. This coverage is the combined "local" coverage of a flat list of all instances of that DU.

  Where <du_name> is [<library name>.]<primary>[(<secondary>)] and secondary name is required only for VHDL.

  <library name>. — (optional) Specifies the library name; if none is specified, then work is assumed. Must be followed by a period (.).

  <primary> — The name of the design unit.

  <secondary> — (optional) The secondary name of the design unit, required for VHDL.

- -duplicates

  (optional) Valid with -toggle switch. Reports all toggle nodes, including alias nodes, using their local names.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -excluded [-pragma | -user] [-nocomment]

  (optional) Includes details on the exclusions in the specified coverage database input file. By default, this option includes both user exclusions and source code pragma exclusions, unless you specify -user or -pragma. The output is structured in DO file command format.

-pragma — When used with the -excluded argument, writes out *only* lines currently being excluded by pragmas.

-user — When used with the -excluded argument, writes out files and lines currently being excluded by the coverage exclude command.

-nocomment — Removes comments from the output, which by default appear in the exclusion report created with -excluded. Comments can be added, modified and removed from the GUI.

- -failed

  (optional) Reports only failed assertions/cover directives.

- -file <filename>

  (optional) Specifies a file name for the report. Default is to write the report to the Transcript window.

  <filename> — A user specified string. Environment variables may be used in the pathname.

- -filter <pattern>

  (optional) Specifies that the command affects only those objects whose name matches the specified pattern. Must be specified with the -assert or -directive argument.

  <pattern> — Specifies the character(s) to be matched in the search. Wildcards are permitted.

- -flat [-primarykey <type | path | value> [-secondarykey <type | path | value>]]

  (optional) Presents coverage items in a text report in a flat view, sorted first by the values specified with -primarykey, and second by those specified by -secondarykey. Only assertions, cover directives, covergroups, toggles and FSMs appear in the flat view of the report. By default, type is the primary sort, and value is secondary.

  -primarykey <type | path | value> — (optional) Specifies the first value for sorting.

  type — (default) Sorts by type of coverage data, asseritons, coverdirectives, covergropus, toggles, and FSMs.

  path — Sorts by the hierarchical path.

  value — Sorts by the value of the coverage items presented in the report.

  -secondarykey <type | path | value> — (optional) Specifies the second value for sorting.

  type — Sorts by type of coverage data, asseritons, coverdirectives, covergropus, toggles, and FSMs.

  path — Sorts by the hierarchical path.

  value — (default) Sorts by the value of the coverage items presented in the report.

- -hidecvginsts

  (optional) Removes all covergroup instances from the report.

- -hidecvginstspi0

  (optional) Removes only the covergroup instances which have per_instance set to 0 from the report.

- -html <input_ucdb> [-code [bcesf[t|x]] [-assert] [-binrhs] [-cvg] [-directive]
  [-details[=abcdefgst]] [-du <path/du]
  ([-dynamic [-servermode |-stopserver] -port <port_no>]]) | [-htmldir <outdir>]
  [-hidecvginsts | -hidecvginstspi0]
  [-instance <path/instance>]
  [-nodu] [-nofec] [-noudp] [-noframes] [-noinstance]
  [-notestplan] [-notimestamp] [-nozeroweights]
  [-showexcluded] [-source] [-stmtaltflow] [-summary]
  [-testhitdata] [-testlist] [-threshL <val>] [-threshH <val>] [-usecnpm] [-verbose]

  (optional) Generate an HTML coverage report on coverage data from a given UCDB file. You can use the -verbose option with -html to enable logging output for each file generated.

  The -html arguments listed below are not compatible with any other vcover report arguments, with the exception of -assert, -cvg, -directive, -binrhs, and-code.

  For descriptions of these arguments, refer to the vcover report argument list.

  input_ucdb — (required) Specifies input UCDB file. Only one input UCDB is allowed.

  -details[=abcdefgst] — (optional) Includes coverage detail pages, which are not included by default. Optionally, you can selectively include only assertions, branch, condition, cover directives, expressions, FSM, covergroups, statements and toggles, or any combination thereof.

  -du <path/du> — (optional) Includes only information for the specified design unit in the HTML report; by default, all design unit information is included. May specify more than one instance by specifying the -du option multiple times in the command, or using wildcards (*) or single character substitutions (?).

  -dynamic [-servermode | -stopserver] -port <port_num>] — (optional, mutually exclusive with -htmldir). The -dynamic argument creates a dynamic HTML report, and opens it in a browser on the machine where the command is given, provided that:

  an environment variable, BROWSER, is set and pointing to a web browser executable (for example, "setenv BROWSER /usr/bin/firefox").

  OR

  the -servermode argument is used:

  The -servermode argument starts the server process on the specified -port <port_num>, and prints the URL which is used to view the report. Temporary files are created in that location, and are deleted when the server is stopped with the -stopserver -port <port_num>.

  -hidecvginsts | -hidecvginstspi0 — (optional) Removes from the HTML report all covergroup instances, or only those having per_instance set to 0.

  -htmldir <outdir> — (optional) Specifies the name of the output directory for the resulting UCDB (default: "covhtmlreport"). Whether you specify an output directory

or the default is used, any file or directory of that name is completely removed prior to report generation to prevent possible stale data. Not compatible with -dynamic.

-instance <path/instance> — (optional) Includes only the specified instance in the HTML report; by default, all instances are included. The instance(s) appears in the report recursively, including any downward hierarchy. May specify more than one instance by specifying the -instance option multiple times in the command, or using wildcards (*) or sinlge character substitutions (?).

-nodu — (optional) Prevents all design unit pages from being generated for the HTML report. By default, design unit information is included in the report.

-nofec / -noudp — (optional) Filters out FEC (-nofec) and UDP (-noudp) condition and expression types from the HTML report.

-noframes — (optional) Avoids generation of a JavaScript-based tree for designs with a large number of design scopes. The report comes up as a single frame containing the top-level summary page and an HTML-only design scope index page is available as a link from the top-level page.

-noinstance — (optional) Prevents all instance pages from appearing in the HTML report. By default, instance information is included in the report.

-notestplan — (optional) Disables generation of testplan HTML pages from merged testplan UCDB files, saving time and disk space during report generation; by default, testplan information is present.

-notimestamp — (optional) Prevents timestamp information from appearing in the report.

-nozeroweights — (optional) Prevents any items with zero weights from appearing in the report, including all zero-weighted coverage items and testplan sections. Has no effect on coverage numbers.

-precision <int> — (optional) Sets the precision for the values displayed in HTML report.

-showexcluded — (optional) Shows the excluded items in the HTML report; by default excluded items are not shown.

-source — (optional) Generates the annotated source. By default, source files are suppressed for HTML. This argument is required in order to access source code related data inside the generated HTML report.

-stmtaltflow — (optional) Allows statement HTML pages to be viewable using Firefox (versions prior to 3.0), Google Chrome, or when viewing via web servers. Without this option, statement pages are only viewable with Firefox version 3.0 and later, Opera, IE7 and later, Safari, and via web servers (only with IE). Use of this option slightly increases size of HTML reports.

-summary — (optional) Includes only the top summary page, the testplan summary page, and the list of tests run in the generated report.

-testhitdata — (optional) Enables generation of a table containing test-bins-hit information. When this argument is used, tables are produced for a merged UCDB (unless merged explicitly with vcover merge -totals) in the HTML report which

contain information regarding which bins were hit by which tests. The table includes bin numbers which are hyperlinked to the relevant test data.

-testlist — (optional) In addition to the test summary page, this argument generates a page of information for each test that was run. By default, these pages are excluded.

-threshL <%> -threshH <val> — (optional) Specifies % of coverage at which colored cells change from red to yellow.

-threshH <%> — (optional) Specifies % of coverage at which colored cells change from yellow to green.

-usecnpm — (optional) uses the value of option.cross_num_print_missing as the display limit for uncovered bins of covergroup cross scopes. By default, all the bins appear.

-verbose — (optional) Prints out the files that are generated by the HTML report generator.

The default output filename is *index.html* in the default directory, *covhtmlreport*.

- -instance <path>

  (optional) Writes out the source file summary coverage data for the selected instance(s). Multiple -instance <path> specifications are allowed. This coverage is the combined "local" coverage of the instance(s). When -recursive is also specified, the local coverage of its children is also displayed.

  <path> — Specifies either an absolute or relative path to the instance(s). On Windows systems the path separator should be a forward slash (/).

- -lang sva | psl | vhdl

  (optional) Specifies assertions of a specific language (SVA, PSL, or VHDL). You may specify multiple languages by using this option multiple times. If -lang is not specified, all three languages are selected.

  sva — System Verilog Assertion language only.

  psl — Property Specification Language only.

  vhdl — VHDL language only.

- -library <libname>

  (optional) Specifies an alternate library. Used when you have packages of the same name in different libraries.

  <libname> — A logical name or pathname to a library.

- -memory

  (optional) Reports a coarse-grain analysis of capacity data for the following SystemVerilog constructs:

  - Classes
  - Queues, dynamic arrays, and associative arrays (QDAS)
  - Assertion and cover directives

  - Covergroups
  - Solver (calls to randomize())

When combined with -cvg and -details, this command reports the detailed memory usage of covergroup. These include the current persistent memory, current transient memory, peak transient memory, and peak time of the following:

  - Per covergroup type
  - Per coverpoint and cross in the type
  - Per covergroup instance (if applicable)
  - Per coverpoint and cross in the instance (if applicable).

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable.

  <path/modelsim.ini> — Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -noannotate

  (optional) Removes source code from the output report. Valid for code coverage only. Not applicable with -xml argument.

- -nocomment | -comment

  (optional) The -nocomment switch prevents comments from appearing in the vcover report output. By default, comments appear in the coverage report.

- -nocollect

  (optional) Prints a report of items whose coverage was not collected. Compatible only with the -file argument, and optionally, -append. Used with -file to print the output to a specified filename, and -append to append the data to the end of the file.

- -nocrossbinsummary

  (optional) Removes covered bins for each coverpoint and cross from the report output.

- -nocvpbinsummary

  (optional) Removes cross bins for each coverpoint and cross from the report output.

- -nodetails

  (optional) Excludes details associated with each coverage item from the output. Details are included by default.

- -noexcludedhits

  (optional) By default, if any excluded items are hit during simulation an “E-hit” notification is displayed in the text report. If “E-hit” is present, it overrides the display of “E” for an

exclusion or “EA”, “ECOP”, and so forth, for various autoexclusions. To disable display of “E-hit” notifications, use the –noexcludedhits option. This has the effect of making all “E” and “EA” type notifications uniformly visible, along with exclusion reasons (if available).

- -nofec

  (optional) Removes FEC expression and condition coverage data from the output report.

- -noignorebin

  (optional) Removes covergroup ignore bins from the report output.

- -nomissing

  (optional) Removes the Misses column from the report output.

  Also removed from the detailed text and HTML reports is the line representing the number of missing bins out of the total number of bins (“missing/total bins:”) line for coverpoints and crosses.

- -noudp

  (optional) If UDP coverage is enabled, it removes UPD expression and condition coverage data from the output report.

- -nozeroweights

  (optional) Prevents any items with zero weights from appearing in the report, including all zero-weighted coverage items and testplan sections. Has no effect on coverage numbers.

- -note <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "note." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information

- -option

  (optional) Includes all covergroup option and type_option values in the report. Unless your covergroup has the "option.per_instance" set to true, only the type_option is included by default. Only applicable to covergroup reports created with the -details argument.

- -package <pkgname>

  (optional) Prints a report on the specified VHDL package body. This argument is equivalent to -du.

  Not a valid option when reporting on a functional coverage database.

  Where <pkgname> = <lib>.<pkg>.

  <lib> — Specifies the library where the package is located, followed by a period (.).

  <pkg> — Specifies the name of the package.

- -portmode {input | output | inout | internal}

  (optional) Prints a report including toggles for only the specified port types.

- -precision <int>

  (optional) Reports on the instance specified with -instance and every included instance, recursively. Can also be used with -details and -totals.

  -depth <n> — (optional) Specifies the maximum recursive depth. A depth of 1 is the same as no recursion.

- -[w]prof=<filename>

  (optional; -wprof and -prof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- -recursive [-depth <n>]

  (optional) Reports on the instance specified with -instance and every included instance, recursively. Can also be used with -details and -totals. Not compatible with -du or -bydu.

  -depth <n> — (optional) Specifies the maximum recursive depth. A depth of 1 is the same as no recursion.

- -samples <spec>

  (optional) Filters the sample count for covergroup types. The sample count is optionally calculated using SVCovergroupSampleInfo, set in the *modelsim.ini* file. This argument filters to include covergroups whose sample count matches the criteria given <spec>. It is a filter for covergroup types only, not instances. If covergroups are found without a sample count, a warning is issued.

  <spec> = {[number] | -le [number] | -ge [number] | [lower]-[upper]}

  [number] — sample count is equal to [number]

  -le [number] — sample count is less than or equal to number

  -ge [number] — sample count is greater than or equal to number

  [lower]-[upper] — (no white space allowed) sample count is in the range lower-upper. There is only 1 -sample option allowed per invocation.

- -select {inputs | outputs | inout | ports | internals}

  (optional) Reports on ports or internal signals. Can be used with the -toggles argument to select type(s) of toggle ports to include in the report.

  inputs — Input ports

  outputs — Output ports

  inout — Inout ports

ports — all ports

internals — internal signals

- -severity info | note | warning | error | failure | fatal

  (optional) Specifies the assertion severity level. When -severity is specified only assertions with the same or higher severity are selected. If not specified, assertions of all severities are selected.

- -showambiguity

  (optional) Displays both minimum and maximum counts for any conflicting toggle data in a UCDB that results from a combined merge (vcover merge command performed with -combine).

- -source <filename>

  (optional) Writes a summary of statement coverage data for a specific source file.

  <filename> — Specifies path and mane to the file. On Windows systems the path separator should be a forward slash (/). Environment variables may be used in the pathname.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

  time — (default) Display Start, End, and Elapsed times.

  Modes

Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover report -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover report -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

kb — Print performance statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the suppress variable in the *modelsim.ini* file to set a permanent default.

- -testattr

  (optional) Display test attributes in the report.

- -testextract <test_name_or_pattern>

  (optional) Display test specific results in the report. Used to combine results from multiple tests. Multiple -testextract arguments can be applied in same command. This argument is compatible with reports generated in plain text and XML formats only, HTML reports are not supported. When using this argument, a header line appears at the top of the report listing test name(s) used to generate the report. Also, the word “hit” appears in place of the count number. UCDB files store only the aggregated coverage counts from all tests, and test-specific numbers can’t be reproduced.

  <test_name_or_pattern> — the test or pattern to extract.

- -totals

  (optional) Writes out a total, recursive summary of whole design, or a specified instance when -instance is used. Useful for tracking changes. Without this argument, the report writes out an instance summary for each of the instances. The report prints only one summary if -totals option is used. Also, when the -totals argument is specified, the alias nodes are not counted.

  A covergroup bin summary is included in the report. It also includes Assertion Passes and/or Successes, Failures, and Attempts, depending on whether vsim -assertcover is used (see vsim -assertcover for more details).

- -toggles

  (optional) Writes out all the toggles in the entire design (not including alias nodes), or under the instance specified by -instance <path>. Valid during simulation, post-processing, and in

vcover. The toggle report generated with this argument is written in the style of reports generated by **toggle report**.

- -top

  (optional) For signals that were added to toggle coverage using vcom or vlog -cover t, -top uses the name of the top-most element of multiple-segment (collapsed) nets. By default the name of the wildcard-matching segment will be used.

- -trend <trend_ucdb> {-bydu | -byinstance | -byplan}
  [-du <duname>] [[-instance | -path] <path>] [-plansection <path>]
  [-recursive [-depth <n>] [-file <output>]
  [-assert] [-cvg] [-code {b|c|e|s|t|f}] [-directive] [-descendants]
  [-above <%>] [-below <%>]
  (([-from <YYYYMMDDHHMMSS | first>] | [-to <YYYYMMDDHHMMSS | last>]) [-numsamples <int>])

  (optional) Plots a text formatted, 2-dimensional graph of a trending database, used for the purpose of analyzing coverage over time for a particular design under test (DUT). Multiple design units, coverage objects, instances, or verification plan sections may be specified in a single command.

  Required arguments for a -trend textual formatted graph:

  {-bydu | -byinstance | -byplan} — Generates reports respectively, according to design units, instances, or according to the verification plan.

  <trend_ucdb> — name of file created with vcover merge -trend command.

  Optional arguments for a -trend textual formatted graph:

  -above <%> — only graphs coverage numbers above the specified percentage.

  -assert — includes only assertions in trend report.

  -below <%> — only graphs coverage numbers below the specified percentage.

  -cvg — includes only covergroups in trend report.

  -code {b|c|e|s|t|f}] — includes only specified code coverage types in trend report.

  -descendants — ensures that the coverage number in the trend report reflects the recursively aggregated coverage for instance hierarchy. Not valid when used with -bydu or -byplan. By default, only the local instance coverage numbers are shown.

  -directive — includes only cover directives in trend report.

  -du <duname> — specifies particular design units to which the report is applied.

  -file <output> — name of the file for text output.

  ([-from <YYYYMMDDHHMMSS| first>] | [-to <YYYYMMDDHHMMSS | last>]) -numsamples <int>] —
  Limits the results to the last X samples, or to the number of X samples starting at a date or backwards ending at a date, or simply a window in time. The -numsamples argument is accepted with either the -from or the -to, not both.

Example: the 3rd of August, 2:30am, to the 5th September at 5pm would be: -from 20130803023000 -to 20130905170000.)

-instance <instance>| -path <path>— specifies particular instances or coverage objects to which the report is applied.

-plansection <path> — specifies particular verification plan sections to which the report is applied.

-recursive [-depth <n>] — generates the report recursively from the specified object(s).

-precision <int> — sets precision for output.

- -trend -csv <trend_ucdb> {-bydu | -byinstance | -byplan}
  [-file <csv_output>] [-assert] [-cvg] [-code {b|c|e|s|t...}] [-directive] [-descendants]
  [-attribute <attr>| -byattribute]
  [-above <%>] [-below <%>] [-precision <int>]
  (([-from <YYYYMMDDHHMMSS | first>] | [-to <YYYYMMDDHHMMSS | last>]) [-numsamples <int>])

  (optional) Plots a 2-dimensional graph, in a comma-separated value format, of a trending database in order to analyze coverage over time for a particular design under test (DUT).

  Required arguments to -trend -csv:

  -csv — creates the graph in comma separated value format.

  <trend_ucdb> — name of file created with vcover merge -trend command.

  {-bydu | -byinstance | -byplan} — One is required. Generates reports respectively, according to design units, instances, or according to the verification plan.

  Optional arguments to -trend -csv:

  -above <%> — only graphs coverage numbers above the specified percentage.

  -assert — includes only assertions in trend report.

  -attribute <name> <value> | -byattribute —
  -attribute includes only the specified attribute in the trend report, where -byattribute eports all attributes that were marked for trending with the coverage attribute command.

  -below <%> — only graphs coverage numbers below the specified percentage.

  -cvg — includes only covergroups in trend report.

  -code {b|c|e|s|t|f}] — includes only specified code coverage types in trend report.

  -descendants — ensures that the coverage number in the trend report reflects the recursively aggregated coverage for instance hierarchy. Not valid when used with -bydu or -byplan. By default, only the local instance coverage numbers are shown.

  -directive — includes only cover directives in trend report.

  -file <CSV_output> — name of the file for CSV output.

  ([-from <YYYYMMDDHHMMSS | first>] | [-to <YYYYMMDDHHMMSS | last>]) -numsamples <int>] —
  Limits the results to the last X samples, or to the number of X samples starting at a

date or backwards ending at a date, or simply a window in time. The -numsamples argument is accepted with either the -from or the -to, not both.
Example: the 3rd of August, 2:30am, to the 5th September at 5pm would be: -from 20130803023000 -to 20130905170000.)

-precision <int> — sets precision for output.

- -trend -html <trend_ucdb> [-htmldir <outdir>]
[-assert] [-cvg] [-code {b|c|e|s|t...}] [-directive] [-descendants]
[-verbose] [-noframes]
[-above <%>] [-below <%>] [-precision <int>] [-summary] [-linegraph]
(([-from <YYYYMMDDHHMMSS | first>] |
[-to <YYYYMMDDHHMMSS | last>]) [-numsamples <int>])

  (optional) Creates graphs for coverage summary, test summary, and linked bin summary, as well as a 2-dimensional graph, in HTML, of a trending database used to analyze coverage over time for a particular design under test (DUT).

  Required arguments to -trend (html version):

  -html — creates the trend graph in HTML format.

  <trend_ucdb> — Required. Name of file created with vcover merge -trend command.

  Optional arguments to -trend (html version):

  -above <%> — only graphs coverage numbers above the specified percentage.

  -assert — includes only assertions in trend report.

  -below <%> — only graphs coverage numbers below the specified percentage.

  -cvg — includes only covergroups in trend report.

  -code {b|c|e|s|t|f}] — includes only specified code coverage types in trend report.

  -descendants — ensures that the coverage number in the trend report reflects the recursively aggregated coverage for instance hierarchy. By default, only the local instance coverage numbers are shown. Not valid when used with -bydu or -byplan.

  -directive — includes only cover directives in trend report.

  ([-from <YYYYMMDDHHMMSS | first>] | [-to <YYYYMMDDHHMMSS | last>]) -numsamples <int>] —
  Limits the results to the last X samples, or to the number of X samples starting at a date or backwards ending at a date, or simply a window in time. The -numsamples argument is accepted with either the -from or the -to, not both.
  Example: the 3rd of August, 2:30am, to the 5th September at 5pm would be: -from 20130803023000 -to 20130905170000.)

  -htmldir <outdir> — name of the directory where HTML output is saved.

  -linegraph — switches the display of the bar graph to a line graph in the HTML report.

  -noframes — avoids generation of the Javascript-based tree, useful for designs with a large number of design scopes.

  -precision <int> — sets precision for output.

-showattrswithcoverage <atr1{,<attr2>][,<attr3>]...[,<attrN>] — displays the user-stored attributes on the generated coverage summary graph, along with the coverage

-summary — generates summary reports only.

-testattr {p|f|w|e|t}— saves user-stored attributes along with the coverage datato be displayed on the same graph as coverage. For example: comparing the number of passed, failed, etc. tests against statement coverage or testplan coverage. Arguments:

p- Number of passed tests

f - Number of fatal-error tests

w - Number of warning tests

e - Number of error tests

t - Number of tests (total)

- -trend -xml <trend_ucdb> {-bydu | -byinstance | -byplan}
  [-file <xml_output>] [-assert] [-cvg] [-code {b|c|e|s|t...}] [-descendants] [-directive]
  [-attribute <name> <value> | -byattribute]
  [-above <%>] [-below <%>] [-precision <int>]
  (([-from <YYYYMMDDHHMMSS | first>] | [-to <YYYYMMDDHHMMSS | last>]) [-numsamples <int>])

  (optional) Plots a 2-dimensional graph, in XML, of a trending database in order to analyze coverage over time for a particular design under test (DUT).

  Required arguments to -trend -xml:

  -xml — creates the graph in XML format.

  <trend_ucdb> — The name of file used as the source for the graph, which has been created with a vcover merge -trend command.

  {-bydu | -byinstance | -byplan} — One of these is required. Generates reports respectively, according to design units, instances, or according to the verification plan.

  Optional arguments to -trend -xml:

  -above <%> — only graphs coverage numbers above the specified percentage.

  -assert — includes only assertions in trend report.

  -attribute <name> <value> | -byattribute — -attribute includes only the specified attribute in the trend report, where -byattribute reports all attributes that were marked for trending with the coverage attribute command.

  -below <%> — only graphs coverage numbers below the specified percentage.

  -cvg — includes only covergroups in trend report.

  -code {b|c|e|s|t|f}] — includes only specified code coverage types in trend report.

-descendants — ensures that the coverage number in the trend report reflects the recursively aggregated coverage for instance hierarchy. By default, only the local instance coverage numbers are shown. Only valid when used with -byinstance.

-directive — includes only cover directives in trend report. Mutually exclusive with -byattribute.

-file <xml_output> — name of the file for XML output. If not used, the file will be saved as in the current directory, as <trend_ucdb>.xml.

-precision <int> — sets precision for output.

- <ucdb_file> | <coverstore>[:test]

  (either UCDB or coverstore is required) The coverage database used to create the report You can specify either a UCDB or a coverstore (created by vsim -coverstore or vcover merge -outputstore command). If you use a coverstore, you can optionally specify a comma separated list of tests.

- -unattemptedimmed

  (optional) Causes the unexecuted (unattempted) immediate assertions to be considered in Total Coverage calculations that are displayed in the coverage report. By default, any unexecuted immediate assertions are not included in the coverage calculations displayed in the coverage report.

- -usecnpm

  (optional) Specifies that the value of SVCrossNumPrintMissingDefault is used in report. By default, all cross bins are displayed in the report.

- -verbose

  (optional) Prints a report listing all the integer values and their counts that an integer toggle encounters during the run. List will include the number of active assertion threads (Active Count) and number of active root threads (Peak Active Count) that have occurred up to the current time.

- -version

  (optional) Returns the version number of UCDB file used to create the report. This argument can not be combined with any other arguments; when present, it invalidates all other arguments.

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -xml

  (optional) Writes out a file-based summary of lines, including file names and line numbers, that have not been executed (zero hits), annotates the source code, and supports the -source and -instance options.

  For covergroups, this argument applies to coverpoint and cross bins only.

  Cannot be used with the -recursive argument.

- -zeros

  (optional) Writes out a file-based summary of lines, including file names and line numbers, that have not been executed (zero hits), annotates the source code, and supports the -source and -instance options.

  For covergroups, this argument applies to coverpoint and cross bins only.

  Cannot be used with the -recursive argument.

  For a detailed report that includes line numbers, use: vcover report -zeros -details.

Examples

- Write a coverage report with all coverage types except toggles :

```
vcover report -code bcefs
```

- Write a top-level summary of the number of instances, statements, branches, hits, signal toggles, and covergroup bins to *myreport.txt*.

```
vcover report -totals -file myreport.txt input.ucdb
```

- Write detailed branch, condition, and statement statistics from *save.ucdb*, without associated source code, to stdout.

```
vcover report -details -code bcs save.ucdb
```

- Write a summary of code coverage for all instances in *save.cov* to stdout.

```
vcover report save.ucdb
```

- Write code coverage details of all instances in *input.ucdb* to *save.cov*. The -details option reports coverage statistics for each statement and branch. Branch coverage statistics will following statement statistics and will be presented in four columns: line, column, true branch count, false branch count.

```
vcover report -details -file save.cov input.ucdb
```

- Write code coverage details of one specific instance to *save.cov*.

```
vcover report -details -instance /top/p -file save.cov input.ucdb
```

- Write a summary of coverage by source file for coverage less than or equal to 90%.

```
vcover report -details -below 90 -file myreport.txt input.ucdb
```

- Write a list of statements with zero coverage to *myzerocov.txt*.

```
vcover report -zeros -file myzerocov.txt input.ucdb
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcover report -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vcover report -stats=time,cmd,msg -stats=none,perf
```

Related Topics

- "Code Coverage"
- coverage save
- "Verification with Assertions and Cover Directives"
- "Verification with Functional Coverage"
- "Analyzing Trends"
- coverage report
- coverage ranktest

vcover stats

The vcover stats command prints to *stdout* summary statistics for previously saved code or functional coverage databases.

It can be invoked within the Questa SIM GUI or at the command line.

vcover stats creates coverage statistics output that is equivalent to the output from this command:

```
vcover report -totals -byinstance
```

This command provides additional information with the -help or -h switch.

Syntax

vcover stats [-32 | -64] [-assert] [-cvg] [-directive] [-code {b | c | e | f | s | t}...] [-codeAll] [-nofec] [-noudp]
[-inputs <pathname>] [-precision <int>] [-memory]
[-modelsimini <path/modelsim.ini>] [-nomissing]
[-[w]prof=<filename>] [-proftick=<integer>] [-stats [=[+ | -]<feature>[,[+ | -]<mode>]]
[-error <msg_number>[,<msg_number>,…]]
[-fatal <msg_number>[,<msg_number>,…]]
[-note <msg_number>[,<msg_number>,…]]
[-suppress <msg_number>[,<msg_number>,…]]
[-warning <msg_number>[,<msg_number>,…]]
{<ucdb_file> | <coverstore>[:test] …}

Arguments

- -32 | -64

  (optional) Specifies whether vcover stats uses the 32- or 64-bit executable. 32 bit is the default. These options apply only to the supported Linux operating systems.

- -assert

  (optional) Reports only assertion coverage data. Includes Assertion Passes, Failures, Successes and Attempts. The summary report generated by this option includes Assertion Passes and/or Successes, Failures, and Attempts, depending on whether vsim -assertcover is used. See vsim -assertcover for detials.

- -code {b | c | e | f | s | t}...

  (optional) Specifies which code coverage statistics to include in the report. By default the report includes statistics for all categories you enabled at compile time.

  This argument is ignored when the UCDB does not contain any code coverage data.

  The characters are as follows:

  b — Include branch statistics.

  c — Include condition statistics.

e — Include expression statistics.

f — Include finite state machine statistics.

s — Include statement statistics.

t — Include toggle statistics.

- -codeAll

  (optional) Applies command to all coverage types. Equivalent to -code bcestf.

- -cvg

  (optional) Reports only covergroup coverage data.

- -directive

  (optional) Reports only directives coverage data.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -fatal <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "fatal" Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- {<ucdb_file> | <coverstore>[:test] …}

  Specifies the UCDB or coverstore for which you want summary statistics, including *.rank* file created by coverage ranktest or vcover ranktest.

  If coverstore is used, you can optionally specify a comma separated list of tests to include in the stats. Multiple pathnames and wildcards are allowed.

- -inputs <pathname>

  (optional) Specifies a text file containing input filenames for which you want to produce statistics.

- -memory

  (optional) Reports a coarse-grain analysis of capacity data for the following SystemVerilog constructs:

  - Classes

  - Queues, dynamic arrays, and associative arrays (QDAS)
  - Assertion and cover directives
  - Covergroups
  - Solver (calls to randomize())

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable.

  <path/modelsim.ini>— Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -nofec

  (optional) Removes FEC coverage data from the summary statistics. Valid for code coverage only. Not applicable with -xml argument.

- -nomissing

  (optional) Removes the Misses column from the output.

  Also removed from the detailed output is the line representing the number of missing bins out of the total number of bins (“missing/total bins:”) line for coverpoints and crosses.

- -note <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "note." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information

- -noudp

  (optional) If UDP coverage is enabled, it removes UPD coverage data from the summary statistics. Valid for code coverage only. Not applicable with -xml argument.

- -precision <int>

  (optional) Sets the decimal precision for printing coverage information. Valid values for <int> are from 0 to 6 and default value is 1 (one). Only the output display is affected by this argument, NOT the contents of the UCDB itself.

- -[w]prof=<filename>

  (optional; -wprof and -prof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

  time — (default) Display Start, End, and Elapsed times.

  Modes

  Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover stats -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover stats -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

  kb — Print performance statistics in Kb units with no auto-scaling.

  list — Display statistics in a Tcl list format when available.

  verbose — Display verbose statistics information when available.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

Edit the suppress variable in the *modelsim.ini* file to set a permanent default.

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

Examples

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcover stats -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vcover stats -stats=time,cmd,msg -stats=none,perf
```

Related Topics

- coverage save
- vcover merge
- "Code Coverage"
- coverage report
- "Verification with Functional Coverage"
- "Verification with Assertions and Cover Directives"
- vcover ranktest
- coverage ranktest

vcover testnames

The vcover testnames command displays the test names in the specified UCDB file. If the UCDB is a merged file, it displays a list of tests in the merged file.

This command is most useful for the -testextract argument of a coverage analyze or vcover report command, because it requires the test name.

(By default, a test's name is the same as the UCDB file, though you can set it explicitly using the -testname argument to the coverage save command.)

This command provides additional information with the -help or -h switch.

Syntax

vcover testnames <file> [-32 | -64] [-tcl] [-modelsimini <path/modelsim.ini>] [-error <msg_number>[,<msg_number>,…]]
[-[w]prof=<filename>] [-proftick=<integer>] [-stats [=[+ | -]<feature>[,[+ | -]<mode>]]
[-note <msg_number>[,<msg_number>,…]]
[-suppress <msg_number>[,<msg_number>,…]]
[-warning <msg_number>[,<msg_number>,…]]

Arguments

- -32 | -64

  (optional) Specifies whether vcover testnames uses the 32- or 64-bit executable. 32 bit is the default. These options apply only to the supported Linux operating systems.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- <file>

  (required) Specifies the UCDB file for which the testnames are displayed.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable.

  <path/modelsim.ini> — Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -note <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "note." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -[w]prof=<filename>

  (optional; -wprof and -prof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

  time — (default) Display Start, End, and Elapsed times.

  Modes

  Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover testnames -stats=cmd+verbose,perf+list. To add or

subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover testnames -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

kb — Print performance statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the suppress variable in the *modelsim.ini* file to set a permanent default.

- -tcl

  (optional) Print attribute information in a tcl format.

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Does not function with internal messages (those without numbers).

  <msg_number> — A number identifying a specific message. Multiple message numbers are specified as a comma separated list.

  Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

Examples

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vcover testnames -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vcover testnames -stats=time,cmd,msg -stats=none,perf
```

Related Topics

- Code Coverage
- Verification Management Browser Window
- Verification Tracker Window
- coverage analyze
- coverage attribute
- coverage exclude
- coverage goal
- coverage ranktest
- coverage report
- coverage save
- coverage tag
- coverage testnames
- coverage weight
- vcover merge
- vcover ranktest
- vcover stats

vdel

This command deletes a design unit from a specified library.

This command provides additional information with the -help switch.

Syntax

vdel [-lib <library_path>] [-modelsimini <path/modelsim.ini>] [-verbose]
{ -all | <primary> [<arch_name>] | -allsystemc | -obj {<object_info>}] | -dpiobj [<object_info>] }

Arguments

- -all

  (optional) Deletes an entire library.

  Caution
  You cannot recover libraries once deleted. You are not prompted for confirmation.

- -allsystemc

  (optional) Deletes all SystemC modules in a design from the working directory.

- -dpiobj [<object_info>]

  (optional) Delete auto-compiled DPI object files.

  <object_info> — Specifies the type of object to remove, as reported by the output of the vdir -obj command. This will take the form of either:

  <compiler> — a string identifying the compiler, such as gcc-3.3.1.

  <platform> — a string identifying the platform.

  <platform-compiler> — a string identifying a compiler/platform pair, such as linux_gcc-3.2.3.

  all — Specifies that all objects should be removed, as reported by the output of the vdir -obj command.

- -lib <library_path>

  (optional) Specifies location of the library that holds the design unit to be deleted. By default, the design unit is deleted from the work library.

  <library_path> — A logical name or pathname of the library.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable.

  <path/modelsim.ini> — Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -obj {<object_info>}

  (optional) removes directories containing SystemC and DPI object files.

  <object_info> — Specifies the type of directory to remove, as reported by the output of the vdir -obj command. This will take the form of either:

  <compiler> — a string identifying the compiler, such as gcc-3.3.1.

  <platform> — a string identifying the platform.

  <platform-compiler> — a string identifying a compiler/platform pair, such as linux_gcc-3.2.3.

  all — Specifies that all directories should be removed, as reported by the output of the vdir -obj command.

- <primary> [<arch_name>]

  (required unless -all is used) Specifies the entity, package, configuration, or module to be deleted.

  This option is not supported for SystemC modules.

  <arch_name> — Specifies the name of an architecture to be deleted. If omitted, all of the architectures for the specified entity are deleted. Invalid for a configuration or a package.

- -verbose

  (optional) Displays progress messages.

Examples

- Delete the work library.

  vdel -all

- Delete the synopsys library.

  vdel -lib synopsys -all

- Delete the entity named xor and all its architectures from the work library.

  vdel xor

- Delete the architecture named behavior of the entity xor from the work library.

  vdel xor behavior

- Delete the package named base from the work library.

  vdel base

vdir

This command lists the contents of a design library and checks the compatibility of a vendor library. If vdir cannot read a vendor-supplied library, the library may not be compatible with Questa SIM.

This command lists SystemC modules that are exported with the SC_MODULE_EXPORT() macro.

This command provides additional information with the -help switch.

Syntax

vdir [-l | [-prop <prop>]] [-r] [-obj] [-all | [-lib <library_name>]] [<design_unit>] [-modelsimini <path/modelsim.ini>]

Arguments

- -all

  (optional) Lists the contents of all libraries listed in the Library section of the active *modelsim.ini* file. Refer to modelsim.ini Variables for more information.

- <design_unit>

  (optional) Indicates the design unit to search for within the specified library. If the design unit is a VHDL entity, its architectures are listed. By default all entities, configurations, modules, packages, and optimized design units in the specified library are listed.

- -l

  (optional) Prints the version of vcom/vlog/sccom with which each design unit was compiled, plus any compilation options used. Also prints the object-code version number that indicates which versions of vcom/vlog/sccom and Questa SIM are compatible.

- -lib <library_name>

  (optional) Specifies the logical name or the pathname of a library to be listed. By default, the contents of the work library are listed.

  <library_name> — A logical name or pathname of a library.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable.

  <path/modelsim.ini> — Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -obj

  (optional) *SystemC only*. Returns the content of the object directories, such as *<work>_sc* or *<work>/_dpi*.

- -prop <prop>

  (optional) Reports on a specified design unit property.

  <prop> — Specifies a Design Unit Property, as listed in Table 2-13. If you do not specify a value for <prop>, an error message is displayed.

Table 2-13. Design Unit Properties

Value of <prop>	Description
archcfg	configuration for arch
body	needs a body
cmpltime	compilation time
default	default options
dir	source directory
dpnd	depends on
entcfg	configuration for entity
fulloptions	Full compile options
inline	module inlined
lock	lock/unlock status
lrm	language standard
mtime	source modified time
name	short name
opcode	opcode format
options	compile options
pdu	preoptimized design unit
root	optimized Verilog design root
src	source file
top	top level model
ver	version string
vlogv	Verilog version
voptv	Verilog optimized version

- -r

  (optional) Prints architecture information for each entity in the output.

Examples

- List the architectures associated with the module named and2 that reside in the default library work.

  vdir -l and2

```
# Library vendor : Model Technology
# Maximum unnamed designs : 3
# MODULE and2
#     Verilog version: <XO@d;_mSdz@12Fz9b]_Z3
#     Version string: 3EdggZ>V3z51fE;>K[51?2
#     Source directory: C:\examples\dataflow_verilog
#     Source modified time: Tue Apr 28 22:48:56 2009
#     HDL source file: gates.v
#     Source file: gates.v
#     Start location: gates.v:18
#     Opcode format: 10.1a; VLOG SE Object version 51
#     Optimized Verilog design root: 1
#     VHDL language standard: 1
#     Compile options: -L mtiAvm -L mtiOvm -L mtiUvm -L mtiUPF
#     Debug Symbol file exists
```

vencrypt

This command encrypts Verilog and SystemVerilog code contained within encryption envelopes. The code is not pre-processed before encryption, so macros and other `directives are unchanged. This allows IP vendors to deliver encrypted IP with undefined macros and `directives.

Upon execution of this command, the filename extension will be changed to *.vp* for Verilog files (*.v* files) and *.svp* for SystemVerilog files (*.sv* files).

If the vencrypt utility processes the file (or files) and does not find any encryption directives it reprocesses the file using the following default encryption:

```
`pragma protect data_method = "aes128-cbc"
`pragma protect key_keyowner = "MTI"
`pragma protect key_keyname = "MGC-DVT-MTI"
`pragma protect key_method = "rsa"
`pragma protect key_block encoding = (enctype = "base64")
`pragma protect begin
```

The vencrypt command must be followed by a compile command – such as vlog – for the design to be compiled.

This command provides additional information with the -help or -hel switch.

Syntax

vencrypt **<filename>** [-d <dirname>] [-e <extension>] [-f <filename>] [-hea <filename>] [-logfile <filename> | -l <filename>] [-o <filename>] [-p <prefix>] [-quiet] [[-stats [=[+ | -]<feature>[,[+ | -]<mode>]]]

Arguments

- <filename>

  (required) Specifies the name of the Verilog source code file to encrypt. One filename is required. Multiple filenames can be entered separated by spaces. Wildcards can be used. Default encryption pragmas will be used, as described above, if no encryption directives are found during processing.

- -d <dirname>

  (optional) Specifies where to save encrypted Verilog files. If no directory is specified, current working directory will be used.

  <dirname> — Specifies the directory to contain the encrypted Verilog or SystemVerilog files. The original file extension (*.v* for Verilog and *.sv* for SystemVerilog) will be preserved.

- -e <extension>

  (optional) Specifies a filename extension.

  <extension> — Any alpha-numeric string.

- -f <filename>

  (optional) Specifies a file with more command line arguments. Allows complex arguments to be reused without retyping. Nesting of -f options is allowed.

  Refer to the section "Argument Files" for more information.

  <filename> — Specifies the name of a file containing command line arguments.

- -hea <filename>

  (optional) Concatenates header information into all design files listed with <filename>. Allows the user to pass a large number of files to the vencrypt utility that do not contain the `pragma protect or `protect information about how to encrypt the file. Saves the user from editing hundreds of files to add in the same `pragma protect to every file.

  <filename> — Specifies an existing file.

- -logfile <filename> | -l <filename>

  (optional) Redirects log output to the file designated by <filename>.

  <filename> — Specifies a file for saving output.

- -o <filename>

  (optional) Combines all encrypted output into a single file.

  <filename> — Specifies a file for saving output.

- -p <prefix>

  (optional) Prepends file names with a prefix.

  <prefix> — Any alpha-numeric string.

- -quiet

  (optional) Disables encryption messages.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd and msg).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg".

  Features

  all — Display all statistics features (cmd, msg, perf). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

cmd — (default) Echo the command line.

msg — (default) Display error and warning summary at the end of command execution.

none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

perf — Display time and memory performance statistics.

time — Display Start, End, and Elapsed times. Has no effect and is ignored.

Modes

Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vencrypt -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vencrypt -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

kb — Print performance statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

Note

vencrypt -quiet disables all default or user-specified -stats features.

Examples

- Insert header information into all design files listed.

  vencrypt -h encrypt_head top.v cache.v gates.v memory.v

  The *encrypt_head* file may look like the following:

  ```
  `pragma protect data_method = "aes128-cbc"
  `pragma protect author = "IP Provider"
  `pragma protect key_keyowner = "MTI", key_method = "rsa"
  `pragma protect key_keyname = "MGC-DVT-MTI"
  `pragma protect begin
  ```

  There is no `pragma protect end expression in the header file, just the header block that starts the encryption. The `pragma protect end expression is implied by the end of the file. For more detailed examples, refer to "Protecting Your Source Code" in the User's Manual.

- Enable the display of message count summary. Echoing of the command line is disabled.

  ```
  vencrypt -stats=msg,-cmd,
  ```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vencrypt -stats=msg,cmd -stats=none,perf
```

Related Topics

- "Protecting Your Source Code" in the User's Manual
- vhencrypt

verror

This command prints a detailed description about a message number. It may also point to additional documentation related to the error.

This command provides additional information with the -help or -h switch.

Syntax

verror [-fmt | -full] <msgNum> ...

verror [-fmt | -full] [-kind <tool>] -all

verror [-kind <tool>] {-pedanticerrors -permissive}

Arguments

- -fmt | -full

  (optional) Specifies the type and amount of information to return.

  -fmt — returns the format string used in the error message.

  -full — returns the format string and complete text associated with the error message.

- [-kind <tool>] -all

  (required when not specifying <msgNum>) Returns information about all messages associated with a specified tool, where <tool> can be one of the following:

aid	hm_entity	mc2com	qverilog
sccom	scgenmod	sdfcomp	sm_entity
vcd2wlf	vcom	vcovkill	vdel
vdir	vencrypt	vgencomp	vish
vlib	vlog	vmake	vmap
vopt	vsim	wlf	wlf2log
wlfman	wlfrecover	xml2ucdb	

- [-kind <tool>] {-pedanticerrors -permissive}

  (optional) Specifies filtering for messages according to either or both of the following:

  -pedanticerrors — display messages that are reported as errors due to adhering to a more strict interpretation of the LRM.

  -permissive — display messages reported as warnings that would be displayed as errors if you use vsim -pedanticerrors.

  where <tool> can be any of the values allowed for the -kind argument.

- <msgNum>

  (required when not specifying -all) Specifies the message number(s) you would like more information about. You can find the message number in messages of the format:

** <Level>: ([<Tool>-[<Group>-]]<MsgNum>) <FormattedMsg>

You can specify <msgNum> any number of times for one verror command in a space-separated list.

Optionally, you can specify the toolname prior to the message number, similar to how it appears in an error message. For example:

```
verror vsim-5003
```

Examples

- If you receive the following message in the transcript:

  ```
  ** Error (vsim-3061) foo.v(22): Too many Verilog port connections.
  ```

  and you would like more information about this message, you would type:

  verror 3061

  and receive the following output:

  ```
  Message # 3061:
  Too many Verilog ports were specified in a mixed VHDL/Verilog
  instantiation. Verify that the correct VHDL/Verilog connection is
  being made and that the number of ports matches.
  [DOC: Questa SIM User's Manual - Mixed VHDL and Verilog Designs
  Chapter]
  ```

vgencomp

Once a Verilog module is compiled into a library, you can use this command to write its equivalent VHDL component declaration to standard output.

Optional switches allow you to generate bit or vl_logic port types; std_logic port types are generated by default.

This command provides additional information with the -help switch.

Syntax

vgencomp [-lib <library_name>] [-b] [-modelsimini <path/modelsim.ini>] [-s] [-v] <module_name>

Arguments

- -lib <library_name>

  (optional) Specifies the working library where the default is to use the work library.

  <library_name> — Specifies the path and name of the working library.

- -b

  (optional) Causes vgencomp to generate bit port types.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable.

  <path/modelsim.ini> — Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -s

  (optional) Used for the explicit declaration of default std_logic port types.

- -v

  (optional) Causes vgencomp to generate vl_logic port types.

- <module_name>

  (required) Specifies the name of the Verilog module to be accessed.

Examples

- This example uses a Verilog module that is compiled into the work library. The module begins as Verilog source code:

```
module top(i1, o1, o2, io1);
   parameter width = 8;
   parameter delay = 4.5;
   parameter filename = "file.in";

   input i1;
   output [7:0] o1;
   output [4:7] o2;
   inout [width-1:0] io1;
endmodule
```

After compiling, vgencomp is invoked on the compiled module:

vgencomp top

and writes the following to stdout:

```
component top
    generic(
      width           : integer := 8;
      delay           : real    := 4.500000;
      filename        : string  := "file.in"
   );
   port(
      i1              : in    std_logic;
      o1              : out   std_logic_vector(7 downto 0);
      o2              : out   std_logic_vector(4 to 7);
      io1             : inout std_logic_vector
   );
end component;
```

vhencrypt

This command encrypts VHDL code contained within encryption envelopes. The code is not compiled before encryption, so dependent packages and design units do not have to exist before encryption.

Upon execution of this command, the *.vhd* filename extension is changed to *.vhdp* and the *.vhdl* filename extension is changed to *.vhdlp*.

If the vhencrypt utility does not find any encryption directives, no output file is produced.

The vhencrypt command must be followed by a compile command – such as vcom – for the design to be compiled.

This command provides additional information with the -help or -hel switch.

Syntax

vhencrypt **<filename>** [-d <dirname>] [-e <extension>] [-f <filename>] [-hea <filename>] [-logfile <filename> | -l <filename>] [-o <filename>] [-p <prefix>] [-quiet] [-stats [=[+ | -]<feature>[,[+ | -]<mode>]]

Arguments

- <filename>

  (required) Specifies the name of the VHDL source code file to encrypt. One filename is required. Multiple filenames can be entered separated by spaces. Wildcards can be used.

- -d <dirname>

  (optional) Specifies where to save encrypted VHDL files. If no directory is specified, the current working directory will be used.

  <dirname> — Specifies the directory to contain the encrypted VHDL files. The original file extension (*.vhd* or *.vhdl*) will be preserved.

- -e <extension>

  (optional) Specifies a filename extension to be applied to the encrypted file.

  <extension> — Any alpha-numeric string.

- -f <filename>

  (optional) Specifies a file with more command line arguments. Allows complex arguments to be reused without retyping. Nesting of -f options is allowed.

  Refer to the section "Argument Files" for more information.

  <filename> — Specifies the name of a file containing command line arguments.

- -hea <filename>

  (optional) Concatenates header information into all design files listed with <filename>. Allows the user to pass a large number of files to the vhencrypt utility that do not contain the encryption information (between the `protect and `protect end directives) about how to

encrypt the file. Saves the user from editing hundreds of files to add the same encryption information into every file.

<filename> — Specifies an existing file.

- -logfile <filename> | -l <filename>

  (optional) Redirects log output to the file designated by <filename>.

  <filename> — Specifies a file for saving output.

- -o <filename>

  (optional) Combines all encrypted output into a single file.

  <filename> — Specifies a file for saving output.

- -p <prefix>

  (optional) Prepends encrypted file names with a prefix.

  <prefix> — Any alpha-numeric string.

- -quiet

  (optional) Disables encryption messages.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd and msg).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg".

  Features

  all — Display all statistics features (cmd, msg, perf). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

  time — Display Start, End, and Elapsed times. Has no effect and is ignored.

  Modes

Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vhencrypt -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vhencrypt -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

kb — Print performance statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

Note

vhencrypt -quiet disables all default or user-specified -stats features.

Examples

- Enable the display of message count summary. Echoing of the command line is disabled.

```
vhencrypt -stats=msg,-cmd,
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vhencrypt -stats=msg,cmd -stats=none,perf
```

Related Topics

- "Protecting Your Source Code" in the User's Manual
- vencrypt

view

This command opens the specified window. If you specify this command without arguments it returns a list of all open windows in the current layout.

When you use the -new argument with the view command Questa SIM will create an additional instance of the specified window type and make it the active window for that type. If multiple instances of a window exist, view will change the active window of that type to the specified window.

Names for windows are generated as follows:

- The first window name (automatically generated without using -new) has the same name as the specified window type. For example, the view wave command will create a wave window named "wave."
- Additional window names, created by using the -new argument, appends an integer to the window type, starting with 1. For example, the next time you use the view wave command, the automatically generated window name will be "wave1." Use the command again and the name will be "wave2," then "wave3," and so forth.
- By default, the designation “- Default” appended to the name of the currently active window.
- You can rename existing windows with the -title argument or create a name for a new window using -new and -title together.

To remove a window, use the noview command.

The view command with one or more options and no window names specified applies the options to the currently open windows. Refer to examples for additional details.

Syntax

view **<window_type>…** [-aliases] [-names] [-new] [-title {New Window Title}]
[-undock {[-icon] [-height <n>] [-width <n>] [-x <n>] [-y <n>]} | -dock]

Arguments

- <window_type>…

  (required) Specifies the window type to view. You do not need to type the full type name (see the examples below); implicit wildcards are accepted; multiple window types are accepted. Available window types are:

assertions	atv	browser	calltree
canalysis	capacity	classgraph	classtree
covergroups	dataflow	details	duranked
exclusions	fcovers	files	fsmlist
fsmview	instance	library	list

locals	memdata	memory	msgviewer
objects	process	profiledetails	project
ranked	runmgr	schematic	source
stackview	structural	structure	tracker
transaction	transcript	uvmdetails	watch
wave			

Not all windows are available with all variants of ModelSim and Questa SIM

When you specify a window type and also use the -new argument, you create a new instance of that window type. You may also specify the window(s) to view when multiple instances of that window type exist (such as wave2). This works only with window names automatically generated by Questa SIM, not with window titles specified with the -title argument.

- -aliases

  (optional) Returns a list of <window_type> aliases.

- -height <n>

  (optional) Specifies the window height in pixels. Can only be used with the -undock switch.

  <n> — Any non-negative integer.

- -icon

  (optional) Toggles the view between window and icon. Can only be used with the -undock switch.

- -names

  (optional) Returns a list of valid <window_type> arguments.

- -new

  (optional) Creates a new instance of the window type specified with the <window_type> argument. New window names are automatically generated by appending an integer to the window type, starting with 1, then incrementing the integer when windows of the same type are created.

- -title {New Window Title}

  (optional) Specifies the window title of the designated window.

  {New Window Title} — Any string. Curly braces are needed for a string containing spaces. Double quotes (" ") can be used in place of braces, for example "New Window Title."

- -dock

  (optional) Docks the specified standalone window into the Main window.

- -undock

  (optional) Opens the specified window as a standalone window, undocked from the Main window.

- -width <n>

  (optional) Specifies the window width in pixels. Can only be used with the -undock switch.

  <n> — Any non-negative integer.

- -x <n>

  (optional) Specifies the window upper-left-hand x-coordinate in pixels. Can only be used with the -undock switch.

  <n> — Any non-negative integer.

- -y <n>

  (optional) Specifies the window upper-left-hand y-coordinate in pixels. Can only be used with the -undock switch.

  <n> — Any non-negative integer.

Examples

- Undock the Wave window from the Main window and makes it a standalone window.

```
view -undock wave
```

- Display an undocked Processes window in the upper left-hand corner of the monitor with a window size of 300 pixels, square.

```
view process -undock -x 0 -y 0 -width 300 -height 300
```

- Display the Watch and Wave windows.

```
view w
```

- Display the Objects and Processes windows.

```
view ob pr
```

- Open a new Wave window with My Wave Window as its title.

```
view -title {My Wave Window} wave
```

- The first command creates a window named ‘wave’. The second command creates a window named ‘wave1’. Its full Tk path is ‘.wave1’. Wave1 is now the active Wave window. Any add wave command would add objects to wave1.

```
view wave
view wave -new
```

- Change the default Wave window back to ‘wave’.

```
view wave
```

- Will override the default Wave window and add *mysig* to wave1.

```
add wave -win .wave1 mysig
```

- Open a new Wave window with "SV_Signals" as its title, then add signals to it.

```
set SV_Signals [view wave -new -title SV_Signals]
add wave -window $SV_Signals /top/mysignals
```

The custom window title "SV_Signals" is saved as a TCL variable, then called using the ‘$’ prefix.

Related Topics

- noview

virtual count

This command reports the number of currently defined virtuals that were not read in using a macro file.

Syntax

virtual count [-kind {implicits | explicits}] [-unsaved]

Arguments

- -kind {implicits | explicits}

  (optional) Reports only a subset of virtuals.

  implicits — virtual signals created internally by the product.

  explicits — virtual signals explicitly created by a user, such as with the virtual signal command.

  Unique abbreviations are accepted.

- -unsaved

  (optional) Reports the count of only those virtuals that have not been saved to a macro file.

Related Topics

- virtual define
- virtual save
- virtual show
- Virtual Objects

virtual define

This command prints to the transcript the definition of the virtual signals, functions, or regions in the form of a command that can be used to re-create the object.

Syntax

virtual define [-kind {implicits | explicits}] <pathname>

Arguments

- -kind {implicits | explicits}

  (optional) Transcripts only a subset of virtuals.

  implicits — virtual signals created internally by the tool.

  explicits — virtual signals explicitly created by a user, such as with the virtual signal command.

  Unique abbreviations are accepted.

- <pathname>

  (required) Specifies the path to the virtual(s) for which you want definitions, where wildcards are allowed.

Examples

- Show the definitions of all the virtuals you have explicitly created.

```
virtual define -kind explicits *
```

Related Topics

- virtual describe
- virtual show
- Virtual Objects

virtual delete

This command removes the matching virtuals.

Syntax

virtual delete [-kind {implicits | explicits}] <pathname>

Arguments

- -kind {implicits | explicits}

  (optional) Removes only a subset of virtuals.

  implicits — virtual signals created internally by the product.

  explicits — virtual signals explicitly created by a user, such as with the virtual signal command.

  Unique abbreviations are accepted.

- <pathname>

  (required) Specifies the path to the virtual(s) you want to delete, where wildcards are allowed.

Examples

- Delete all of the virtuals you have explicitly created.

```
virtual delete -kind explicits *
```

Related Topics

- virtual signal
- virtual function
- Virtual Objects

virtual describe

This command prints to the transcript a complete description of the data type of one or more virtual signals.

Similar to the existing describe command.

Syntax

virtual describe [-kind {implicits | explicits}] <pathname>

Arguments

- -kind {implicits | explicits}

  (optional) Transcripts only a subset of virtuals.

  implicits — virtual signals created internally by the product.

  explicits — virtual signals explicitly created by a user, such as with the virtual signal command.

  Unique abbreviations are accepted.

- <pathname>

  (required) Specifies the path to the virtual(s) for which you want descriptions, where wildcards are allowed.

Examples

- Describe the data type of all virtuals you have explicitly created.

```
virtual describe -kind explicits *
```

Related Topics

- virtual define
- virtual show
- Virtual Objects

virtual expand

This command prints to the transcript a list of all the non-virtual objects contained in the specified virtual signal(s).

You can use this to create a list of arguments for a command that does not accept or understand virtual signals.

Syntax

virtual expand [-base] <pathname> ...

Arguments

- -base

  (optional) Outputs the root signal parent in place of a subelement. For example:

  ```
  vcd add [virtual expand -base myVirtualSignal]
  ```

  the resulting command after substitution would be:

  ```
  vcd add signala signalb signalc
  ```

- <pathname>

  (required) Specifies the path to the signals and virtual signals to expand, where wildcards are allowed and you can specify any number of paths.

Examples

- Add the elements of a virtual signal to the VCD file.

  In the Tcl language, the square brackets specify that the enclosed command should be executed first ("virtual expand …"), then the result substituted into the surrounding command.

  ```
  vcd add [virtual expand myVirtualSignal]
  ```

  Therefore, if *myVirtualSignal* is a concatenation of *signala*, *signalb.rec1* and *signalc(5 downto 3)*, the resulting command after substitution would be:

  ```
  vcd add signala signalb.rec1 {signalc(5 downto 3)}
  ```

  The slice of *signalc* is enclosed in curly braces, because it contains spaces.

Related Topics

- virtual signal
- Virtual Objects

virtual function

This command creates a new signal, known only by the GUI (not the kernel), that consists of logical operations on existing signals and simulation time, as described in <expressionString>.

It cannot handle bit selects and slices of Verilog registers. Please see *Syntax and Conventions* for more details on syntax.

If the virtual function references more than a single scalar signal, it will display as an expandable object in the Wave and Objects windows. The children correspond to the inputs of the virtual function. This allows the function to be "expanded" in the Wave window to see the values of each of the input waveforms, which could be useful when using virtual functions to compare two signal values.

Virtual functions can also be used to gate the List window display.

Note

The virtual function and virtual signal commands are interchangeable. The product will keep track of whether you've created a signal or a function with the commands and maintain them appropriately. We document both commands because the virtual save, virtual describe, and virtual define commands will reference your virtual objects using the correct command.

Syntax

virtual function [-env <path>] [-install <path>] [-delay <time> <unit>] {<expressionString>} <name>

Arguments

Arguments for virtual function are the same as those for virtual signal, except for the contents of the expression string.

- -env <path>

  (optional) Specifies a hierarchical context for the signal names in <expressionString> so they don't all have to be full paths.

  <path> — Specifies a relative path to the signal(s).

- -install <path>

  (optional) Causes the newly-created signal to become a child of the specified region. If -install is not specified, the newly-created signal becomes a child of the nearest common ancestor of all objects appearing in <expressionString>. If the expression references more than one WLF file (dataset), the virtual signal will automatically be placed in region virtuals:/Functions.

  <path> — Specifies a relative path to the signal(s). On Windows systems the path separator should be a forward slash (/).

- -delay <time> <unit>

  (optional) Specifies a value by which the virtual function will be delayed. You can use negative values to look forward in time. Refer to the examples below for more details.

  <time> — Specified as an integer or decimal number. Current simulation units are the default unless specifying <unit>.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid VHDL time units are: fs, ps, ns, us, ms, sec, min, and hr. You must enclose <time> and <unit> within curly braces ({}).

- {<expressionString>}

  (required) A text string expression, enclosed in curly braces ({ }) using the GUI_expression_format.

- <name>

  (required) The name you define for the virtual signal.

  Case is ignored unless installed in a Verilog region.

  Use alpha, numeric, and underscore characters only, unless you are using VHDL extended identifier notation.

  If using VHDL extended identifier notation, <name> needs to be quoted with double quotes (" ") or with curly braces ({ }).

Examples

- Create a signal */chip/section1/clk_n* that is the inverse of */chip/section1/clk*.

```
virtual function { not /chip/section1/clk } clk_n
```

- Create a std_logic_vector equivalent of a Verilog register *rega* and installs it as */chip/rega_slv*.

```
virtual function -install /chip { (std_logic_vector) chip.vlog.rega
} rega_slv
```

- Create a boolean signal */chip/addr_eq_fab* that is true when */chip/addr[11:0]* is equal to hex "fab", and false otherwise. It is acceptable to mix VHDL signal path notation with Verilog part-select notation.

```
virtual function { /chip/addr[11:0] == 0xfab } addr_eq_fab
```

- Create a signal that is high only during times when signal */chip/siga* of the gate-level version of the design does not match */chip/siga* of the rtl version of the design. Because there is no common design region for the inputs to the expression, *siga_diff* is installed in region *virtuals:/Functions*. The virtual function *siga_diff* can be added to the Wave window, and when expanded will show the two original signals that are being compared.

```
virtual function { gate:/chip/siga XOR rtl:/chip/siga } siga_diff
```

- Create a virtual signal consisting of the logical "AND" function of */top/signalA* with */top/signalB*, and delays it by 10 ns.

```
virtual function -delay {10 ns} {/top/signalA AND /top/signalB}
myDelayAandB
```

- Create a one-bit signal *outbus_diff* which is non-zero during times when any bit of */chip/outbus* in the gate-level version doesn't match the corresponding bit in the rtl version.

  This expression uses the "OR-reduction" operator, which takes the logical OR of all the bits of the vector argument.

```
virtual function { | (gate:/chip/outbus XOR rtl:/chip/outbus) }
outbus_diff
```

Commands fully compatible with virtual functions

add log and log	delete	describe
examine	find	restart
searchlog	show	
add list	add wave	checkpoint and restore
down and up	left and right	search

Commands not compatible with virtual functions

drivers	force	noforce
vcd add	when	
check contention add	check contention config	check contention off
check float add	check float config	check float off
check stable on	check stable off	power add
power report	power reset	toggle add
toggle reset	toggle report	

Related Topics

- virtual count
- virtual define
- virtual delete
- virtual describe
- virtual expand
- virtual hide
- virtual log
- Virtual Objects
- virtual nohide
- virtual nolog
- virtual region
- virtual save
- virtual show
- virtual signal
- virtual type

virtual hide

This command causes the specified real or virtual signals to not be displayed in the Objects window. This is used when you want to replace an expanded bus with a user-defined bus.

You make the signals reappear using the virtual nohide command.

Syntax

virtual hide {{[-kind {implicits | explicits}] | [-region <path>]} <pattern>

Arguments

- -kind {implicits | explicits}

  (optional) Hides only a subset of virtuals.

  implicits — virtual signals created internally by the tool.

  explicits — virtual signals explicitly created by a user, such as with the virtual signal command.

  Unique abbreviations are accepted.

- -region <path>

  (optional) Specifies a region of design space in which to look for the signal names.

  <path> — Specifies an absolute or relative path to the signal(s). On Windows systems the path separator should be a forward slash (/).

- <pattern>

  (required) Indicates which signal names or wildcard patterns should be used in finding the signals to hide, where wildcards are allowed and you can specify any number of names or patterns.

Related Topics

- virtual nohide
- Virtual Objects

virtual log

This command causes the simulation-mode dependent signals of the specified virtual signals to be logged by the kernel.

If wildcard patterns are used, it will also log any normal signals found, unless the -only option is used. You unlog the signals using the virtual nolog command.

Syntax

virtual log {[-kind {implicits | explicits}] | [-region <path>]} [-recursive] [-only] [-in] [-out] [-inout] [-internal] [-ports] <pattern>

Arguments

- -kind {implicits | explicits}

  (optional) Logs only a subset of virtuals.

  implicits — virtual signals created internally by the tool.

  explicits — virtual signals explicitly created by a user, such as with the virtual signal command.

  Unique abbreviations are accepted.

- -region <path>

  (optional) Specifies a region of design space in which to look for signals to log.

  <path> — Specifies an absolute or relative path to the signal(s). On Windows systems the path separator should be a forward slash (/).

- -recursive

  (optional) Specifies that the scope of the search is to descend recursively into subregions. If omitted, the search is limited to the selected region.

- -only

  (optional) Specify that only virtual signals (as opposed to all signals) found by a <pattern> containing a wildcard should be logged.

- -in

  (optional) Specifies that the kernel log data for ports of mode IN whose names match the specification.

- -out

  (optional) Specifies that the kernel log data for ports of mode OUT whose names match the specification.

- -inout

  (optional) Specifies that the kernel log data for ports of mode INOUT whose names match the specification.

- -internal

  (optional) Specifies that the kernel log data for internal (non-port) objects whose names match the specification.

- -ports

  (optional) Specifies that the kernel log data for all ports.

- <pattern>

  (required) Indicates which signal names or wildcard patterns should be used in finding the signals to log, where you can specify any number of names or wildcard patterns.

Related Topics

- Virtual Objects
- virtual nolog

virtual nohide

This command reverses the effect of a virtual hide command, causing the specified real or virtual signals to reappear the Objects window.

Syntax

virtual nohide {[-kind {implicits | explicits}] | [-region <path>]} <pattern>

Arguments

- -kind {implicits | explicits}

  (optional) Unhides only a subset of virtuals.

  implicits — virtual signals created internally by the tool.

  explicits — virtual signals explicitly created by a user, such as with the virtual signal command.

  Unique abbreviations are accepted.

- -region <path>

  (optional) Specifies a region of design space in which to look for the signal names.

  <path> — Specifies an absolute or relative path to the signal(s). On Windows systems the path separator should be a forward slash (/).

- <pattern>

  (required) Indicates which signal names or wildcard patterns should be used in finding the signals to hide, where wildcards are allowed and you can specify any number of names or patterns.

Related Topics

- virtual hide
- Virtual Objects

virtual nolog

This command reverses the effect of a virtual log command. It causes the simulation-dependent signals of the specified virtual signals to be excluded ("unlogged") by the kernel.

If wildcard patterns are used, it will also unlog any normal signals found, unless the -only option is used.

Syntax

virtual nolog {[-kind {implicits | explicits}] | [-region <path>]} [-recursive] [-only] [-in] [-out] [-inout] [-internal] [-ports] <pattern>

Arguments

- -kind {implicits | explicits}

  (optional) Excludes only a subset of virtuals.

  implicits — virtual signals created internally by the tool.

  explicits — virtual signals explicitly created by a user, such as with the virtual signal command.

  Unique abbreviations are accepted.

- -region <path>

  (optional) Specifies a region of design space in which to look for signals to unlog.

  <path> — Specifies an absolute or relative path to the signal(s). On Windows systems the path separator should be a forward slash (/).

- -recursive

  (optional) Specifies that the scope of the search is to descend recursively into subregions. If omitted, the search is limited to the selected region.

- -only

  (optional) Specify that only virtual signals (as opposed to all signals) found by a <pattern> containing a wildcard should be unlogged.

- -in

  (optional) Specifies that the kernel exclude data for ports of mode IN whose names match the specification.

- -out

  (optional) Specifies that the kernel exclude data for ports of mode OUT whose names match the specification.

- -inout

  (optional) Specifies that the kernel exclude data for ports of mode INOUT whose names match the specification.

- -internal

  (optional) Specifies that the kernel exclude data for internal (non-port) objects whose names match the specification.

- -ports

  (optional) Specifies that the kernel exclude data for all ports.

- <pattern>

  (required) Indicates which signal names or wildcard patterns should be used in finding the signals to unlog, where wildcards are allowed and you can specify any number of names or patterns.

Related Topics

- virtual log
- Virtual Objects

virtual region

This command creates a new user-defined design hierarchy region.

Note

Virtual regions cannot be used in the when command.

Syntax

virtual region <parentPath> <regionName>

Arguments

- <parentPath>

  (required) The full path to the region that will become the parent of the new region.
- <regionName>

  (required) The name you want for the new region.

Related Topics

- virtual function
- virtual signal
- Virtual Objects

virtual save

This command saves the definitions of virtuals to a file named virtual.do in the current directory.

Syntax

virtual save [-kind {implicits | explicits}] [-append] [<filename>]

Arguments

- -kind {implicits | explicits}

  (optional) Saves only a subset of virtuals.

  implicits — virtual signals created internally by the tool.

  explicits — virtual signals explicitly created by a user, such as with the virtual signal command.

  Unique abbreviations are accepted.

- -append

  (optional) Specifies to save only virtuals that are not already saved or weren't read in from a macro file. These unsaved virtuals are then appended to the specified or default file.

- <filename>

  (optional) The name of the file containing the definitions. If you don't specify <filename>, the default virtual filename (*virtuals.do*) will be used. You can specify a different default in the *pref.tcl* file.

Related Topics

- virtual count
- Virtual Objects

virtual show

This command lists the full path names of all explicitly defined virtuals.

Syntax

virtual show [-kind {implicits | explicits}]

Arguments

- -kind {implicits | explicits}

  (optional) Lists only a subset of virtuals.

  implicits — virtual signals created internally by the tool.

  explicits — virtual signals explicitly created by a user, such as with the virtual signal command.

  Unique abbreviations are accepted.

Related Topics

- virtual define
- virtual describe
- Virtual Objects

virtual signal

This command creates a new signal, known only by the GUI (not the kernel), that consists of concatenations of signals and subelements as specified in <expressionString>.

It cannot handle bit selects and slices of Verilog registers. Please see Concatenation of Signals or Subelements for more details on syntax.

Note

The virtual function and virtual signal commands are interchangeable. The product will keep track of whether you've created a signal or a function with the commands and maintain them appropriately. We document both commands because the virtual save, virtual describe, and virtual define commands will reference your virtual objects using the correct command.

Syntax

virtual signal [-env <path>] [-install <path>] [-delay <time> <unit>] {<expressionString>} <name>

Arguments

- -env <path>

  (optional) Specifies a hierarchical context for the signal names in <expressionString> so they don't all have to be full paths.

  <path> — Specifies a relative path to the signal(s). On Windows systems the path separator should be a forward slash (/).

- -install <path>

  (optional) Causes the newly-created signal to become a child of the specified region. If -install is not specified, the newly-created signal becomes a child of the nearest common ancestor of all objects appearing in <expressionString>. If the expression references more than one WLF file (dataset), the virtual signal will automatically be placed in region virtuals:/Signals.

  <path> — Specifies a relative path to the signal(s). On Windows systems the path separator should be a forward slash (/).

- -delay <time> <unit>

  (optional) Specifies a value by which the virtual function will be delayed. You can use negative values to look forward in time. Refer to the examples below for more details.

  <time> — Specified as an integer or decimal number. Current simulation units are the default unless specifying <unit>.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid VHDL time units are: fs, ps, ns, us, ms, sec, min, and hr. You must enclose <time> and <unit> within curly braces ({}).

- {<expressionString>}

  (required) A text string expression, enclosed in curly braces ({ }) using the GUI_expression_format.

- <name>

  (required) The name you define for the virtual signal.

  Case is ignored unless installed in a Verilog region.

  Use alpha, numeric, and underscore characters only, unless you are using VHDL extended identifier notation.

  If using VHDL extended identifier notation, <name> needs to be quoted with double quotes (" ") or with curly braces ({ }).

Examples

- Reconstruct a bus *sim:/chip/alu/a(4 downto 0)*, using VHDL notation, assuming that *a_ii* are all scalars of the same type.

```
virtual signal -env sim:/chip/alu { (concat_range (4 downto 0))(a_04
& a_03 & a_02 & a_01 & a_00) } a
```

- Reconstruct a bus *sim:chip.alu.a[4:0]*, using Verilog notation. Note that the concatenation notation starts with "&{" rather than "{".

```
virtual signal -env sim:chip.alu
{ (concat_range [4:0])&{a_04, a_03, a_02, a_01, a_00} } a
```

- Create a signal *sim:/testbench/stuff* which is a record type with three fields corresponding to the three specified signals. The example assumes */chipa/mode* is of type integer, */chipa/alu/a* is of type std_logic_vector, and */chipa/decode/inst* is a user-defined enumeration.

```
virtual signal -install sim:/testbench
{ /chipa/alu/a(19 downto 13) & /chipa/decode/inst & /chipa/mode }
stuff
```

- Create a virtual signal that is the same as */top/signalA* except it is delayed by 10 ps.

```
virtual signal -delay {10 ps} {/top/signalA} myDelayedSignalA
```

- Create a three-bit signal, *chip.address_mode*, as an alias to the specified bits.

```
virtual signal { chip.instruction[23:21] } address_mode
```

- Concatenate signals *a*, *b*, and *c* with the literal constant ‘000’.

```
virtual signal {a & b & c & 3'b000} myextendedbus
```

- Add three missing bits to the bus *num*, creates a virtual signal *fullbus*, and then adds that signal to the Wave window.

```
virtual signal {num & "000"} fullbus
add wave -unsigned fullbus
```

- Reconstruct a bus that was fragmented by synthesis and is missing the lower three bits. Note that you would have to type in the actual bit names (for example, num28, num27, and so on) represented by the … in the syntax above.

```
virtual signal { num31 & num30 & num29 & ... & num4 & num3 & "000" }
fullbus
add wave -unsigned fullbus
```

- Create a two-bit signal (with an enumerated type) based on the results of the subexpressions. For example, if *aold* equals *anew*, then the first bit is true (1). Alternatively, if *bold* does not equal *bnew*, the second bit is false (0). Each subexpression is evaluated independently.

```
virtual signal {(aold == anew) & (bold == bnew)} myequalityvector
```

- Create signal *newbus* that is a concatenation of bus1 (bit-reversed) and bus2[7:4] (bit-reversed). Assuming bus1 has indices running 7 downto 0, the result will be newbus[11:0] with the upper 8 bits being bus1[0:7] and the lower 4 bits being bus2[4:7]. See Concatenation Directives for further details.

```
virtual signal {(concat_reverse)(bus1 & bus2[7:4])} newbus
```

Commands fully compatible with virtual signals

add list	add log or log	add wave
delete	describe	examine
find	force and noforce	restart
searchlog	show	
checkpoint and restore	down and up	left and right
search		

Commands compatible with virtual signals using [virtual expand <signal>]

drivers	vcd add	
check contention add	check contention config	check contention off
check float add	check float config	check float off
check stable on	check stable off	
power add	power report	power reset
toggle add	toggle reset	toggle report

Commands not currently compatible with virtual signals

- when

Related Topics

- virtual count
- virtual describe
- virtual log
- virtual region
- virtual function
- virtual define
- virtual expand
- virtual nohide
- virtual save
- virtual type
- virtual delete
- virtual hide
- virtual nolog
- virtual show
- Virtual Objects

virtual type

This command creates a new enumerated type known only by the GUI, not the kernel. Virtual types are used to convert signal values to character strings. The command works with signed integer values up to 64 bits.

Virtual types cannot be used in the when command.

Note

If you are using SystemVerilog, you can also convert signal values to character strings using associative arrays in your code. See the SystemVerilog LRM for more information.

Syntax

virtual type -delete <name> | {<list_of_strings>} <name>

Arguments

- -delete <name>

  (Required if not defining a type.) Deletes a previously defined virtual type.

  <name> — The name you gave the virtual type when you originally defined it. .

- {<list_of_strings>}

  (Required if -delete is not used.) A list of values and their associated character strings. Values can be expressed in decimal or based notation and can include "don’t-cares" (see examples below). Three kinds of based notation are supported: Verilog, VHDL, and C-language styles. The values are interpreted without regard to the size of the bus to be mapped. Bus widths up to 64 bits are supported.

  If the string contains spaces the string must be enclosed in quotation marks (“‘) If they contain special characters square brackets, curly braces, backslashes…), they need to be quoted within curly braces.

  See the examples below for further syntax.

- <name>

  (Required if -delete is not used.) The user-defined name of the virtual type. Case is not ignored. Use alpha, numeric, and underscore characters only, unless you are using VHDL extended identifier notation. If using VHDL extended identifier notation, <name> needs to be quoted with double quotes (" ") or with curly braces ({ }).

Examples

- Using positional notation, associates each string with an enumeration index, starting at zero and increasing by one in the positive direction.When *myConvertedSignal* is displayed in the Wave, List, or Objects window, the string "state0" will appear when *mysignal* == 0, "state1" when *mysignal* == 1, "state2" when *mysignal* == 2, and so on.

```
virtual type {state0 state1 state2 state3} mystateType
virtual function {(mystateType)mysignal} myConvertedSignal
add wave myConvertedSignal
```

- Use sparse mapping of bus values to alphanumeric strings for an 8-bit, one-hot encoding. It shows the variety of syntax that can be used for values. The value "default" has special meaning and corresponds to any value not explicitly specified.

```
virtual type {{0 NULL_STATE} {1 st1} {2 st2} {0x04 st3} {16'h08 st4} \
             {'h10 st5} {16#20 st6} {0b01000000 st7} {0x80 st8} \
             {default BAD_STATE}} myMappedType
virtual function {(myMappedType)mybus} myConvertedBus
add wave myConvertedBus
```

- Delete the virtual type "mystateType".

```
virtual type -delete mystateType
```

- Create a virtual type that includes "don’t-cares" (the ‘-’ character).

```
virtual type {{0x01-- add}{0x02-- sub}{default bad}} mydecodetype
```

- Create a virtual type using a mask for "don’t-cares." The middle field is the mask, and the mask should have bits set to 1 for the bits that are don't care.

```
virtual type {{0x0100 0xff add}{0x0200 0xff sub}{default bad}}
mydecodetype
```

Related Topics

- virtual function
- Virtual Objects

vlib

This command creates a design library. You must use vlib rather than operating system commands to create a library directory or index file.

If the specified library already exists as a valid Questa SIM library, the vlib command will exit with a warning message without touching the library.

This command provides additional information with the -help switch.

Syntax

vlib -help

vlib [-short | -dos | -long | -unix] [-format { 1 | 3 | 4 }]
[-type {directory | archive | flat}]
[{-lock | -unlock} <design_unit>] [-locklib | -unlocklib] [-unnamed_designs <value>]
[-compress | -nocompress]
<library_name>

Arguments

- -compress | -nocompress

  (optional) Defines whether some compiled results are stored in the library in a compressed form.

  -compress — Compression occurs, producing smaller libraries. However, this can slow down your subsequent executions of the vopt command.

  -nocompress — (default) No compression is made on the libraries.

- -dos

  (optional) Specifies that subdirectories in a library have names that are compatible with DOS. Not recommended if you use the vmake utility.

- -format { 1 | 3 | 4 }

  (optional) Prepares a library for conversion to be compatible with a previous release, by altering the _info file.

  1 — allows you to convert a library to be compatible with the 6.2 series and earlier.

  3 — allows you to convert a library to be compatible with the 6.3 series and newer.

  4 — allows you to convert a library to be compatible with the 10.2 series and newer.

  The usage flow is:

```
\\1) Using a current release of the simulator, run:
        vlib -format 1 current_lib
        vcom -refresh -work current_lib
\\    to prepare current_lib for conversion back to a 6.2 release
\\
\\2) Using a 6.2 release of the simulator, run:
        vcom -refresh -work current_lib
\\    to refresh current_lib for use with the previous release
```

- -long

  (optional) Interchangeable with the -unix argument.

- {-lock | -unlock} <design_unit>

  (optional) Locks an existing design unit so it cannot be recompiled or refreshed. The -unlock switch reverses this action. File permissions are not affected by these switches.

- -locklib | -unlocklib

  (optional) Locks a complete library so that compilation cannot target the library and the library cannot be refreshed. The -unlocklib switch reverses this action. File permissions are not affected by these switches.

  Refer to the section "Creating Locked Libraries for Multiple-User Simulation Environments" in the User's Manual for a scenario that uses these switches.

- -short

  (optional) Interchangeable with the -dos argument.

- -type {directory | archive | flat}

  (optional) Specifies the type of library you want to create.

  directory — directory-based, legacy library. Use this option when working in a flow requiring the vmake command.

  archive — archive library (replaces vlib -archive option).

  flat — (default) condensed library without design unit directories.

- -unix

  (optional) Specifies that subdirectories in a library may have long file names that are NOT compatible with DOS.

  On by default for Questa.

- -unnamed_designs <value>

  (optional) Specifies how many unnamed, optimized versions of a design the vopt command will save within the library. Once <value> is reached, vopt deletes the oldest unnamed, optimized version. By default, the maximum number of "unnamed" designs ("_opt[number]") is set to 3.

- <library_name>

  (required) Specifies the pathname of the library to be created.

Examples

- Create the design library *design*. You can define a logical name for the library using the vmap command or by adding a line to the library section of the *modelsim.ini* file that is located in the same directory.

```
vlib design
```

- Create the design library *uut* and specifies that any design units compiled into the library are created as archives.

```
vlib -type archive uut
```

vlog

The vlog command compiles Verilog source code and SystemVerilog extensions into a specified working library (or to the work library by default). Compressed SystemVerilog source files (those compressed with zlib) are accepted.

The vlog command may be invoked from within Questa SIM or from the operating system command prompt. It may also be invoked during simulation.

Compiled libraries are major-version dependent. When moving between major versions, you have to refresh compiled libraries using the -refresh argument to vlog. This is not true for minor versions (letter releases).

All arguments to the vlog command are case sensitive: -WORK and -work are not equivalent.

SystemVerilog requires that the default behavior of the vlog command is to treat each Verilog design file listed on the command line as a separate compilation unit. To treat multiple files listed within a single command line as a single compilation unit, use either the vlog -mfcu argument or the MultiFileCompilationUnit *modelsim.ini* file variable.

This command provides additional information with the -help switch.

Syntax

vlog [options] <filename> [<filename> ...]

[options]:

[-32 | -64] [-93]

[-addpragmaprefix <prefix>]
[-ccflags <“compileopts”>] [-compat] [-compile_uselibs[=<directory_name>]] [-constimmedassert | -noconstimmedassert] [-convertallparams] [+cover[=<spec>]] [-covercells | -nocovercells] [-coverdeglitch {“<n> <time_unit>”}] [-coverenhanced] [-coverexcludedefault | -nocoverexcludedefault] [-coverfec | -nocoverfec] [-coveropt <opt_level>] [-coverrec | -nocoverrec] [-coverreportcancelled] [-coverudp | -nocoverudp] [-cuname] [-cuautoname=[file | du]]

[+define+<macro_name>[=<macro_text>]] [-deglitchalways | -nodeglitchalways] [+delay_mode_distributed] [+delay_mode_path] [+delay_mode_unit] [+delay_mode_zero] [-dpicppinstall <[gcc|g++] version>] [-dpicpppath <pathname>] [-dpiforceheader] [-dpiheader <filename>]

[-E <filename>] [-Edebug <filename>] [-enumfirstinit] [-Epretty <filename>] [-error <msg_number>[,<msg_number>,…]] [-extendedtogglemode 1|2|3]

[(-F | -file | -f) <filename>] [-fecudpeffort {1 | 2 | 3}] [-force_refresh <design_unit>] [-fsmimplicittrans | -nofsmimplicittrans] [-fsmmultitrans] [-fsmresettrans | -nofsmresettrans] [-fsmsingle | -nofsmsingle] [-fsmverbose[b | t | w]] [-fsmxassign | -nofsmxassign]

[-gen_xml <design_unit> <filename>]

[-hazards]

[-ignorepragmaprefix <prefix>] [+incdir+<directory>] [-incr | -noincr]
[+initmem[=<spec>][+{0 | 1 | X | Z}]]
[+initreg[=<spec>][+{0 | 1 | X | Z}]] [-isymfile] [+iterevaluation]

[-L <libname>] [-Lf <libname>] [+libcell | +nolibcell] [+libext+<suffix>]
[-libmap <pathname>] [-libmap_verbose] [+librescan] [-line <number>]
[-lint]
[-logfile <filename> | -l <filename>] [-lowercasepragma] [-lowercasepslpragma]
[-lrmclassinit]

[+maxdelays] [+mindelays] [-mixedansiports] [-mixedsvvh [b | s | v]]
[-mfcu[=macro] | -sfcu] [-modelsimini <path/modelsim.ini>]

[-nocovercells] [-nocoverfec] [-nocovershort] [-nocoverudp] [-nocreatelib] [-nodbgsym]
[-nodebug[=ports | =pli | =ports+pli]] [-noexcludeternary <design_unit>]
[-noForceUnsignedToVhdlInteger] [-nologo] [-nopsl] [+nospecify]
[-note <msg_number>[,<msg_number>,…]] [-novopt] [+notimingchecks]
[-novtblfixup] [+nowarn<CODE>] [-nowarn <category_number>]
[-nosparse] [+num_opt_cell_conds+<value>]

[-optionset <optionset_name>] [-outf <filename>]
[-override_timescale[=]|[ ]<time_unit> / <time_precision>] [-O0 | -O1 | -O4 | -O5]

[-pedanticerrors] [-permissive] [-permit_defunct_sv] [-printinfilenames[=<filename>]]
[-proftick=<integer>] [-[w]prof=<filename>] [+protect[=<filename>]]
[-pslext] [-pslfile <filename>]

[-quiet]

[-R [<simargs>]] [-refresh]

[-s] [-scdpiheader <filename>] [-sfcu] [-skipprotected] [-skipprotectedmodule]
[-skipsynthoffregion] [-smartdbgsym] [-source]
[-stats [=[+ | -]<feature>[,[+ | -]<mode>]]
[-suppress <msg_number>[,<msg_number>,…]] [-sv]
[-svext=[+|-]<extension>[,[+|-]<extension>]…]
[-svfilesuffix=<extension>[,<extension>...] <filename>]
[-svinputport=net | var | relaxed] [-svpkgcasesens]
[-sv05compat] [-sv09compat] [-sv12compat]

[-timescale[=]|[ ]<time_units>/<time_precision>] [-togglecountlimit <int>]
[-togglewidthlimit <int>] [-toggleportsonly] [+typdelays]

[-u]

[-v <library_file>] [-version] [-vlog01compat] [-vlog95compat] [-vmake]
[-vopt | -novopt]

[-warning <msg_number>[,<msg_number>,…]] [-warning error] [-warnrbw]
[-work <library_name>] [-writetoplevels <fileName>]

[-y <library_directory>]

Using vlog for Access Control for Visibility During Optimization

The following optional arguments are replacements for the vlog +acc argument and its values.

vlog [-assertdebug [=[-]<accesscodes>]] [-bitscalars [=[-]<accesscodes>]] [-cellaccess [=[-]<accesscodes>]] [-floatparameters] [-fsmdebug [=[-]<accesscodes>]] [-linedebug [=[-]<accesscodes>]] [-nosparse] [-primitiveaccess [=[-]<accesscodes>]] [-randmetastable [=[-]<accesscodes>]] [-systfoverride [=[-]<accesscodes>]] <filename> [<filename> ...]

Note

Use of the +acc argument with vlog is not recommended for the following reasons:

- It may reduce simulation speed.
- You should use the access debug arguments for the vopt command instead. This will simplify compilation and optimization flows and make them easier to maintain.
- The +acc argument and its values have been replaced by more specific arguments to the vlog and vopt commands. The vlog +acc argument is supported but deprecated.

Arguments

- -32 | -64

  Specifies whether vlog uses the 32- or 64-bit executable, where -32 is the default.

  These options apply only to the supported Linux operating systems.

  These options override the MTI_VCO_MODE environment variable, which applies only to executables used from the *<install_dir>/bin/* directory. Therefore, these options are ignored if you run vlog from an *<install_dir>/<platform>/* directory.

  You can only specify these options on the command line, therefore they are not recognized as part of a file used with the -f switch.

- -93

  (optional) Specifies that the VHDL interface to Verilog modules use VHDL 1076-1993 extended identifiers to preserve case in Verilog identifiers that contain uppercase letters.

- -addpragmaprefix <prefix>

  (optional) Enables recognition of pragmas with a user specified prefix. If this argument is not specified, pragmas are treated as comments.

  All regular synthesis and coverage pragmas are honored.

  <prefix> — Specifies a user defined string where the default is no string, indicated by quotation marks (“”).

  You may also set this with the AddPragmaPrefix variable in the vlog section of the *modelsim.ini* file.

- -assertdebug [=[-]<accesscodes>]

  (optional) Allow to debug SVA/PSL objects when used with vsim -assertdebug.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

- -bitscalars [=[-]<accesscodes>]

  (optional) Preserve access to net bits.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

- -ccflags <“compileopts”>

  Specify in quotes all the C/C++ compiler options for the DPI auto compile flow. Multiple occurrences are supported.

- -ccwarn [on | off | verbose | strict]

  Echoes warnings generated during C/C++ source file compilation to stdout.

  on — (default) compiles with the -Wreturn-type, -Wimplicit, -Wuninitialized, and -Wmissing-declarations gcc options.

  off — compiles without any warning options.

  verbose —compiles with the -Wall gcc option.

  strict — compiles with the -Werror gcc option.

- -cellaccess [=[-]<accesscodes>]

  (optional) Preserve access to cell internal objects.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

- -compat

  (optional) Disables optimizations that result in different event ordering than Verilog-XL.

  Questa SIM Verilog generally duplicates Verilog-XL event ordering, but there are cases where it is inefficient to do so. Using this option does not help you find event order dependencies, but it allows you to ignore them. Keep in mind that this option does not account for all event order discrepancies, and that using this option may degrade performance. Refer to "Event Ordering in Verilog Designs" for additional information.

- -compile_uselibs[=<directory_name>]

  (optional) Locates source files specified in a `uselib directive (Refer to "Verilog-XL uselib Compiler Directive"), compiles those files into automatically created libraries, and updates the *modelsim.ini* file with the logical mappings to the new libraries. If a *directory_name* is not specified, Questa SIM uses the name specified in the MTI_USELIB_DIR environment variable. If that variable is not set, Questa SIM creates the directory *mti_uselibs* in the current working directory.

- -constimmedassert

  (optional) Displays immediate assertions with constant expressions in the GUI, in reports, and in the UCDB. By default, immediate assertions with constant expressions are displayed in the GUI, in reports, and in the UCDB. Use this switch only if the -noconstimmedassert switch has been used previously, or if the ShowConstantImmediateAsserts variable in the vlog section of the *modelsim.ini* file is set to 0 (off).

- -noconstimmedassert

  (optional) Turns off the display of immediate assertions with constant expressions in the GUI, in reports, and in the UCDB. By default, immediate assertions with constant expressions are displayed. You may also set the ShowConstantImmediateAsserts variable in the vlog section of the *modelsim.ini* file to 0 (off).

- -convertallparams

  (optional) Enables converting parameters not defined in ANSI style to VHDL generics of type std_logic_vector, bit_vector, std_logic, vl_logic, vl_logic_vector, and bit.

- +cover[=<spec>]

  (optional) Enables various coverage statistics collection on all design units compiled in the current compiler run.

  Consider using the +cover or +nocover arguments to vopt instead, which you can use to specify precise design units and regions to be instrumented (or uninstrumented) for coverage.

  The +cover argument with no "=<spec>" designation is equivalent to "+cover=bcesft".

  <spec> — one or more of the following characters:

  b — Collect branch statistics.

  c — Collect condition statistics. Collects only FEC, unless -coverudp is specified.

e — Collect expression statistics, Collects only FEC statistics, unless -coverudp is specified.

s — Collect statement statistics.

t — Collect toggle statistics. Overridden if ‘x’ is specified elsewhere.

x — Collect extended toggle statistics (Refer to “Toggle Coverage” for details). This takes precedence, if ‘t’ is specified elsewhere.

f — Collect Finite State Machine statistics.

See -coveropt <opt_level> argument to override the default level of optimization for coverage for a particular compilation run.

Refer to “Rules for Applying Coverage with cover and nocover” for information on how potentially conflicting commands are resolved.

- -covercells

  (optional) Enables code coverage of modules defined by ‘celldefine and ‘endcelldefine compiler directives, or compiled with the -v or -y arguments. Can be used to override the CoverCells compiler control variable in the *modelsim.ini* file.

- -coverdeglitch {“<n> <time_unit>”}

  (optional) Enables deglitching of statement, branch, condition and expression coverage in combinational unclocked process/always blocks, concurrent (VHDL) and continuous (SV) assignments, where:

  [n] — A string specifying the number of time units.

  <time_unit> — (required if “n” is anything other than “0”) fs, ps, ns, us, ms, or sec.

  Quotes (“”) are required if a space exists between the two arguments (for example, 2ps or “2 ps”).

  If a process or a continuous assignment is evaluated more than once during any period of length <period>, only the last execution during that period is added into the coverage data for that process/continuous assignment. If a deglitch period of zero is specified, only the last delta cycle pass is counted toward the coverage data. For a new pass to be counted, it must occur at a time greater than the previous pass plus the deglitch period. You can customize the default behavior with the CoverDeglitchOn and CoverDeglitchPeriod variables in the *modelsim.ini* file.

- -coverenhanced

  (optional) Enables non-critical functionality which might change the appearance or content of coverage metrics. This argument only has an effect in letter releases (10.0a, 10.0b, and so on). In major releases (10.0, 10.1, and so on), all coverage enhancements present in previous letter release streams are enabled by default, and -coverenhanced is no longer necessary to enable these enhancements. Bug fixes important to the correctness of coverage numbers are always enabled by default, with no need for -coverenhanced. Since the exact nature of -coverenhanced varies from release to release, the details of the enhancements it enables are

present in the product release notes rather than in the Command Reference. For these details, search the release notes using the string "coverenhanced".

- -coverfec

  (optional) Enables focused expression coverage (FEC) for coverage collection. By default, FEC coverage statistics are disabled for collection. You can customize the default behavior with the CoverFEC variable in the *modelsim.ini* file.

- -coverexcludedefault

  (optional) Excludes Verilog/SV code coverage data collection from the default branch in case statements. All forms of code coverage collection in the default case statement will be excluded. It also excludes the allfalse branch (if any) associated with the case statement.

- -coveropt <opt_level>

  (optional) Overrides the default level of optimization for the current run only. <opt_level> designates the optimization level, as follows:

  1 — Turns off all optimizations that affect coverage reports.

  2 — Allows optimizations that provide large performance improvements by invoking sequential processes only when the data changes. This setting may result in major reductions in coverage counts.

  3 — (default) Allows all optimizations in 2, and allows optimizations that may remove some statements. Also allows constant propagation and VHDL subprogram inlining.

  4 — Allows all optimizations in 2 and 3, and allows optimizations that may remove major regions of code by changing assignments to built-ins or removing unused signals. It also changes Verilog gates to continuous assignments and optimizes Verilog expressions. Allows VHDL subprogram inlining. Allows VHDL flip-flop recognition.

  The default optimization level is 3. You can edit the CoverOpt variable in the *modelsim.ini* file to change the default.

- -coverrec

  (optional) Enables rapid expression coverage (REC) for coverage collection. By default, REC coverage statistics are enabled for collection. You can customize the default behavior with the CoverREC variable in the *modelsim.ini* file.

- -coverreportcancelled

  (optional) Enables code coverage reporting of branch conditions that have been optimized away due to a static or null condition. The line of code is labeled EA in the hits column of the Source Window and EBCS in the hits column of a Coverage Report. You can also set this with the CoverReportCancelled *modelsim.ini* variable.

- -coverudp

  (optional) Enables UDP coverage for coverage collection. By default, FEC coverage statistics are enabled for collection. You can customize the default behavior with the CoverUDP variable in the *modelsim.ini* file.

- -cuname

  (optional) Used only in conjunction with -mfcu. The -cuname names the compilation unit being created by vlog. The named compilation unit can then be specified on the vsim command line, along with the <top> design unit. The purpose of doing so is to force elaboration of specified compilation unit package, thereby forcing elaboration of a necessary 'bind' statement within that compilation unit that would otherwise not be elaborated. An example of the necessary commands is:

  ```
  vlog -cuname pkg_name -mfcu file1.sv file2.sv
  vsim top pkg_name
  ```

  You need to do this only in cases where you have a 'bind' statement in a module that might otherwise not be elaborated, because no module in the design depends on that compilation unit. In other words, if a module that depends on that compilation unit exists, you don't need to force the elaboration, for it occurs automatically. Also, if you are using qverilog to compile and simulate the design, this binding issue is handled properly automatically.

- -cuautoname=[file | du]

  (optional) Specifies the method for naming $unit library entries.

  file — (default) Base the name on first file in on the command line.

  du — Base the name on the first design unit following items found in the $unit scope. This option is useful for cases where you have multiple vlog command lines that specify the same file as the first entry.

- +define+<macro_name>[=<macro_text>]

  Allows you to define a macro from the command line that is equivalent to the following compiler directive:

  ```
  `define <macro_name> <macro_text>
  ```

  Optionally, you can specify more than one macro with a single +define. For example:

  ```
  vlog +define+one=r1+two=r2+three=r3 test.v
  ```

  A command line macro overrides a macro of the same name defined with the `define compiler directive.

- -deglitchalways | -nodeglitchalways

  Controls the behavior related to zero-delay oscillations among always_comb and always @* cominatorial logic blocks, as well as regular always blocks, that produce glitches on the variables they write.

  -deglitchalways — (default) Reduces the incidents of zero delay oscillations among the affected blocks.

-nodeglitchalways — Disables the functionality. A side effect of this behavior is that time zero races involving the glitch-producing always blocks may resolve in a different order.

- +delay_mode_distributed

  (optional) Disables path delays in favor of distributed delays. Refer to "Delay Modes" for details.

- +delay_mode_path

  (optional) Sets distributed delays to zero in favor of using path delays.

- +delay_mode_unit

  (optional) Sets path delays to zero and non-zero distributed delays to one time unit.

- +delay_mode_zero

  (optional) Sets path delays and distributed delays to zero.

- -dpicppinstall <[gcc|g++] version>

  (optional) Specifies the version of the desired GNU compiler supported and distributed by Mentor Graphics for the DPI exportwrapper compilation.

  <[gcc|g++] version> — The version number of the GNU compiler to use. For example:

  ```
  vsim -dpicppinstall 4.5.0
  ```

- -dpicpppath <pathname>

  (optional) Specifies an explicit location to a gcc compiler for use with automatically compiled DPI C/C++ files. Ensures that the argument points directly to the compiler executable. This overrides the DpiCppPath variable in the *modelsim.ini* file.

- -dpiforceheader

  (optional) Forces the generation of a DPI header file even if it will be empty of function prototypes.

- -dpiheader <filename>

  (optional) Generates a header file that may then be included in C source code for DPI import functions. Refer to "DPI Use Flow" for additional information.

- -E <filename>

  (optional) Captures text processed by the Verilog parser after preprocessing has occurred and copies that text to an output file. This includes text read from source files specified by using the -v or -y argument.

  <filename> — Specifies a name for the debugging output file. Wildcards are not allowed.

  Generally, preprocessing consists of the following compiler directives: `ifdef, `else, `elsif, `endif, `ifndef, `define, `undef, `include.

The `line directive attempts to preserve line numbers, file names, and level in the output file (per the 1800-2009 LRM). White space is usually preserved, but sometimes it may be deleted or added to the output file.

- -Edebug <filename>

  (optional) Captures text processed by the Verilog parser after preprocessing has occurred and copies that text to a debugging output file.

  <filename> — Specifies a name for the debugging output file. Wildcards are not allowed.

  Generally, preprocessing consists of the following compiler directives: `ifdef, `else, `elsif, `endif, `ifndef, `define, `undef, `include. The file is a concatenation of source files with `include expanded. The file can be compiled and then used to find errors in the original source files. The `line directive attempts to preserve line numbers and file names in the output file. White space is usually preserved, but sometimes it may be deleted or added to the output file.

- -enumfirstinit

  (optional) Initializes enum variables in SystemVerilog using the leftmost value as the default. You must also use the argument with the vsim command in order to implement this initialization behavior. Specify the EnumBaseInit variable as 0 in the *modelsim.ini* file to set this as a permanent default.

- -Epretty <filename>

  (optional) Captures text processed by the Verilog parser after preprocessing has occurred, performs some formatting for better readability, and copies that text to an output file, <filename>.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information.

- -extendedtogglemode 1|2|3

  (optional) Changes the level of support for extended toggles. The levels of support are:

  1 — 0L->1H & 1H->0L & any one 'Z' transition (to/from 'Z')

  2 — 0L->1H & 1H->0L & one transition to 'Z' & one transition from 'Z'

  3 — 0L->1H & 1H->0L & all 'Z' transitions

  Edit the ExtendedToggleMode variable in the *modelsim.ini* file to set a permanent default.

- (-F | -file | -f) <filename>

  (optional) -f, -file and -F: each specifies an argument file with more command-line arguments, allowing complex argument strings to be reused without retyping. Nesting of -F, -f and -file commands is allowed. Allows gzipped input files.

With -F only: relative file names and paths within the arguments file <filename> are prefixed with the path of the arguments file when lookup with relative path fails. Refer to the section "Argument Files" for more information.

- -fecudpeffort {1 | 2 | 3}

  (optional) The recommended option to set the level of effort for FEC or UDP. Levels are:

  1 — (default) (low) Only small expressions or conditions are considered for coverage.

  2 — (medium) Bigger expressions and conditions considered for coverage.

  3 — (high) Very large expressions and conditions considered for coverage.

  Increasing the effort level includes more expressions/conditions for coverage, but also increases the compile time. You can also enable this by setting the FecUdpEffort variable in the *modelsim.ini* file.

- -floatparameters

  Do not hold parameter values constant during optimization. This enables use of the vsim -g/G arguments on the affected parameters. Using this argument causes finite state machines (FSMs) containing these floating parameters not to be recognized.

- -force_refresh <design_unit>

  (optional) Forces the refresh of all specified design units. By default, the work library is updated; use -work <library_name>, in conjunction with -force_refresh, to update a different library (for example, vlog -work <your_lib_name> -force_refresh).

  When the compiler refreshes a design unit, it checks each dependency to ensure its source has not been changed and recompiled. Sometimes the dependency checking algorithm changes from release to release. This can lead to false errors during the integrity checks performed by the -refresh argument. An example of such a message follows:

  ```
  ** Error: (vsim-13) Recompile /u/test/dware/dware_61e_beta.dwpackages
  because /home/users/questasim/../synopsys.attributes has changed.
  ```

  The -force_refresh argument forces the refresh of the design unit, overriding any dependency checking errors encountered by the -refresh argument.

  A more conservative approach to working around -refresh dependency checks is to recompile the source code, if it is available.

- -fsmdebug [=[-]<accesscodes>]

  (optional) Allow access to finite state machines for debugging.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

- -fsmimplicittrans | -nofsmimplicittrans

  (optional) Toggles recognition of implied same state transitions. This setting is off by default.

- -fsmmultitrans

  (optional) Enables detection and reporting of multi-state transitions when used with the +cover=f argument for vlog or vopt. Another term for this is FSM sequence coverage.

- -fsmresettrans | -nofsmresettrans

  (optional) Toggles recognition of synchronouos or asynchronous reset transitions.

  This includes/excludes reset transitions in coverage results.

  This setting is on by default.

- -fsmsingle | -nofsmsingle

  (optional) Toggles the recognition of VHDL FSMs where the current state variable of type std_logic, bit, boolean, or single-bit std_logic_vector/bit_vector and Verilog single-bit FSMs. This setting is off by default.

- -fsmverbose[b | t | w]

  (optional) Provides information about FSMs detected, including state reachability analysis.

  b — displays only basic information.

  t — displays a transition table in addition to the basic information.

  w — displays any warning messages in addition to the basic information.

  When you do not specify an argument, this switch reports all information similar to:

```
# ** Note: (vlog-1947)   FSM RECOGNITION INFO
#      Fsm detected in : ../fpu/rtl/vhdl/serial_mul.vhd
#      Current State Variable : s_state :
../fpu/rtl/vhdl/serial_mul.vhd(76)
#      Clock : clk_i
#      Reset States are: { waiting , busy }
#      State Set is : { busy , waiting }
#      Transition table is
#         -------------------------------------------
#         busy      =>   waiting Line : (114 => 114)
#         busy      =>   busy    Line : (111 => 111)
#         waiting   =>   waiting Line : (120 => 120) (114 => 114)
#         waiting   =>   busy    Line : (111 => 111)
#         -------------------------------------------
```

  When you do not specify this switch, you will receive a message similar to:

```
# ** Note: (vlog-143) Detected '1' FSM/s in design unit
'serial_mul.rtl'.
```

- -fsmxassign | -nofsmxassign

  (optional) Toggles recognition of finite state machines (FSMs) containing X assignment. This option is used to detect FSMs if current state variable or next state variable has been assigned "X" value in a "case" statement. FSMs containing X-assign are otherwise not detectable. This setting is on by default.

- -gen_xml <design_unit> <filename>

  (optional) Produces an XML-tagged file containing the interface definition of the specified module. This option requires a two-step process where you must 1) compile <filename> into a library with vlog (without -gen_xml) then 2) execute vlog with the -gen_xml switch, for example:

```
vlib work
vlog counter.v
vlog -gen_xml counter counter.xml
```

- -hazards

  (optional) Detects event order hazards involving simultaneous reading and writing of the same register in concurrently executing processes. You must also specify this argument when you simulate the design with vsim. Refer to "Hazard Detection" for more details.

Note

Enabling -hazards implicitly enables the -compat argument. As a result, using this argument may affect your simulation results.

- -ignorepragmaprefix <prefix>

  (optional) Directs vlog to ignore pragmas with the specified prefixname. All affected pragmas will be treated as regular comments. Edit the IgnorePragmaPrefix *modelsim.ini* variable to set a permanent default.

  <prefix> — Specifies a user defined string.

- +incdir+<directory>

  (optional) Specifies directories to search for files included with `include compiler directives. By default, the current directory is searched first and then the directories specified by the +incdir options in the order they appear on the command line. You may specify multiple +incdir options as well as multiple directories separated by "+" in a single +incdir option.

  When you compile a file that imports the OVM package shipped with Questa SIM, you do not need to specify +incdir+ to the OVM directory. For example, if your *.sv* file contains:

```
import ovm_pkg::*;
`include "ovm_macros.svh"
```

  The vlog command will automatically add the +incdir switch for you, issuing the following note:

```
** Note: (vlog-2286) Using implicit
+incdir+<install_dir>/<ovm_dir>/src
from import ovm_pkg
```

  However, if you are using an open version of the OVM you will need to explicitly specify that directory with the +incdir switch.

  Note that a package import of the built-in UVM library now adds an implicit +incdir entry that is derived from package import location.

- -incr

  (optional) Performs an incremental compilation. Compiles only code that has changed. For example, if you change only one module in a file containing several modules, only the changed module will be recompiled. Note however that if the compile options change, all modules are recompiled, regardless of whether you use vlog -incr or not.

- +initmem[=<spec>][+{0 | 1 | X | Z}]

  (optional) Enables the initialization of memories.

  <spec> — (optional) identifies the types to be initialized.

  If you do not specify this option, vlog initializes fixed-size arrays of all these types, where fixed-size arrays may have any number of packed or unpacked dimensions.

  <spec> can be one or more of the following:

  r — register/logic, integer, or time types (four-state integral types).

  b — bit, int, shortint, longint, or byte types (two-state integral types).

  e — enum types.

  You must also add the enum's base type to the initialization specification. If you choose static initialization for an enum type variable with value 0, 1, X, or Z, the simulator assigns that value to the variable, whether it is a valid value or not. If you choose random initialization for an enum type variable, the simulator generates a random number and uses the (random_number % num_valid_enum_values)th entry of the enum literals to initialize it.

  +{0 | 1 | X | Z} — (optional) specifies the initialization value to use for all the elements of a memory. For example, the +initmem+1 option initializes "reg [7:0] m" to value 'hff. For two-state datatypes, X and Z will map to 0.

  If you do not specify the value, initmem will randomly initialize this memory to a 2-state value. The seed for randomization can then be controlled by vsim command line option +initmem+<seed>, where <seed> is a 32-bit number.

  These options do not result in scheduling events at time 0. The specified memories are initialized to these values. So, if you set these memories to the same values that +initmem initialized them to, events will not be generated. The exception is that of sequential UDPs that are initialized in the NBA region of time 0.

  This argument initializes static variables in any scope (package, $unit, module, interface, generate, program, task, function). However, it does not affect:

  - automatic variables
  - dynamic variables
  - members of dynamic variables
  - artificially generated variables, such as #randstate#

  In case +initmem options are specified to vlog and vopt command line , the priority of the specifications are as follows:

```
1) design unit or instance specific options specified during vopt –
   for example, vopt ... +initmem+1+top.foo
2) options specified during vlog — for example, vlog ... +initmem+0
3) global options specified during vopt — vopt … +initmem+Z
```

This argument will not override any variable declaration assignment, such as:

```
reg r = 1'b0
```

- +initreg[=<spec>][+{0 | 1 | X | Z}]

  (optional) Enables you to initialize registers.

  <spec> — identifies the types to be initialized.

  If you do not specify this option, vlog initializes variables of all these types.

  <spec> can be one or more of the following:

  r — register/logic, integer, or time types (four-state integral types).

  Notifier registers are not initialized by the +initreg option. To detect that a register is a notifier, timing checks must be present, which means you cannot compile with the +nospecify or +notimingchecks arguments. However, if you want to remove timing checks but still detect notifier registers, use vsim +notimingchecks or vsim +nospecify. You can also do this is by using `ifdef to remove timing checks.

  b — bit, int, shortint, longint, or byte types (two-state integral types).

  e — enum types.

  You must also add the enum's base type to the initialization specification. If you choose static initialization for an enum type variable with value 0, 1, X, or Z, the simulator assigns that value to the variable, whether it is a valid value or not. If you choose random initialization for an enum type variable, the simulator generates a random number and uses the (random_number % num_valid_enum_values)th entry of the enum literals to initialize it.

  u — sequential UDPs. Valid if UDP coverage is enabled.

  If a sequential UDP contains an "initial" statement, that initial value overrides all +initreg-related functionality. For other sequential UDPs, the +initreg option takes effect as described for regular variables. In case a sequential UDP does not contain an "initial" statement, and it wasn't compiled with +initreg in effect, the UDPs initial value will be taken from its instantiating parent scope (provided that scope has +initreg options in effect).

  +{0 | 1 | X | Z} — (optional) specifies the initialization value to use for all the bits of a register. For example, +initreg+1 option initializes "reg [7:0] r" to value 'hff. For two-state datatypes, X and Z will map to 0.

  If you do not specify the value, initreg will randomly initialize this register to a 2-state value. The seed for randomization can then be controlled by vsim command line option +initreg+<seed>, where <seed> is a 32-bit number.

  These options do not result in scheduling events at time 0. The specified registers are initialized to these values. So, if you set these registers to the same values that

+initreg initialized them to, events will not be generated. The exception is that of sequential UDPs that are initialized in the NBA region of time 0.

This argument initializes static variables in any scope (package, $unit, module, interface, generate, program, task, function). However, it does not affect:

- automatic variables
- dynamic variables
- members of dynamic variables
- artificially generated variables, such as #randstate#

In case +initreg options are specified to vlog and vopt command line , the priority of the specifications are as follows:

```
1) design unit or instance specific options specified during vopt –
   for example, vopt ... +initreg+1+top.foo
2) options specified during vlog — for example, vlog ... +initreg+0
3) global options specified during vopt — vopt … +initreg+Z
```

This argument will not override any variable declaration assignment, such as:

```
reg r = 1'b0
```

- -isymfile

  Generates a complete list of all imported tasks and functions (TFs). Used with DPI to determine all imported TFs that are expected by Questa SIM.

- +iterevaluation

  (default) Enable an iterative evaluation mechanism on optimized gate-level cells with feedback loops.

- -logfile <filename> | -l <filename>

  (optional) Generates a log file of the compile.

  -logfile <filename> — Saves transcript data to <filename>. Can be abbreviated to -l <filename>. Overrides the default transcript file creation set with the the TranscriptFile or BatchTranscriptFile *modelsim.ini* variables. You can also specify “stdout” or “stderr” for <filename>.

- -L <libname>

  (optional) Searches the specified resource library for precompiled modules. The library search options you specify here must also be specified when you run the **vsim** command. See also the LibrarySearchPath variable and Verilog Resource Libraries in the User’s Manual.

- -Lf <libname>

  (optional) Same as -L, but the specified library is searched before any `uselib directives. (Refer to “Library Usage” and “Verilog-XL Compatible Compiler Arguments” for more information.).

- +libcell | +nolibcell

  +libcell — (optional) Treats all modules found and compiled by source library search as though they contained a ‘celldefine compiler directive, thus marking them as cells (refer to the -v and -y arguments of vlog, which enable source library search). Using the +libcell argument matches historical behavior of Verilog-XL with respect to source library search.

  +nolibcell — (default) Disables treating all modules found and compiled by source library search as though they contained a ‘celldefine compiler directive. That is, this argument restores the default library search behavior if you have changed it using the +libcell | +nolibcell argument.

Note

By default, wildcard logging and code coverage exclude cells. For more information, refer to the -nocovercells and -covercells arguments of vlog and to the description of wildcard logging performed by the log command.

- +libext+<suffix>

  (optional) Works in conjunction with the -y option. Specifies file extensions for the files in a source library directory. By default, the compiler searches for files without extensions. If you specify the +libext argument, then the compiler will search for a file with the suffix appended to an unresolved name. You may specify only one +libext option, but it may contain multiple suffixes separated by the plus character (+). The extensions are tried in the order you specify them with the +libext argument.

- -libmap <pathname>

  (optional) Specifies a Verilog 2001 library map file. You can omit this argument by placing the library map file as the first option on the vlog invocation (for example, *vlog top.map top.v top_cfg.v*). You can use the vlog -mfcu argument to compile macros for all files in a given testbench. Any macros already defined before the -libmap argument appears are still defined for use by the -libmap files.

- -libmap_verbose

  (optional) Displays library map pattern matching information during compilation. Use to troubleshoot problems with matching filename patterns in a library file.

- +librescan

  (optional) Scans libraries in command-line order for all unresolved modules.

- -line <number>

  (optional) Starts the compiler on the specified line in the Verilog source file. By default, the compiler starts at the beginning of the file.

- -linedebug [=[-]<accesscodes>]

  (optional) Allow line debugging.

=<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

c — Preserve connectivity access on selected object.

r — Preserve read access on selected object.

w — Preserve write access on selected object.

- -lint

  (optional) Issues warnings on the following lint-style static checks:

  - when Module ports are NULL.
  - when assigning to an input port.
  - when referencing undeclared variables/nets in an instantiation.

  This switch generates additional array bounds-checking code, which can slow down simulation, to check for the following:

  - index warnings for dynamic arrays
  - when an index for a Verilog unpacked variable array reference is out of bounds.

  The warnings are reported as WARNING[8]. You can also enable this option using the Show_Lint variable in the *modelsim.ini* file.

- -lowercasepragma

  (optional) Accepts only lower case pragmas in Verilog source files. You can also enable this feature by setting the AcceptLowerCasePragmaOnly variable in the *modelsim.ini* file.

- -lowercasepslpragma

  Forces the Verilog compiler to accept only lower case (embedded) PSL pragmas.

- -lrmclassinit

  Changes initialization behavior to match the SystemVerilog specification (per IEEE Std 1800-2007) where all superclass properties will be initialized before any subclass properties.

- +maxdelays

  (optional) Selects maximum delays from the "min:typ:max" expressions. If preferred, you can defer delay selection until simulation time by specifying the same option to the simulator.

- +mindelays

  (optional) Selects minimum delays from the "min:typ:max" expressions. If preferred, you can defer delay selection until simulation time by specifying the same option to the simulator.

v — treats all scalars/vectors in the package as VHDL vl_logic/vl_logic_vector

- -mfcu[=macro]

  (optional) Instructs the compiler to treat all files within a compilation command line as a single compilation unit. The default behavior is to treat each file listed in a command as a separate compilation unit, per the SystemVerilog standard. Prior versions concatenated the contents of the multiple files into a single compilation unit by default. When specified, the =macro modifier enables the visibility of macro definitions across different files.

  You can use -mfcu to compile macros for all files in a given testbench. Any macros already defined before the -libmap argument appears are still defined for use by the -libmap files.

  You can also enable this option (without the =macro functionality) using the MultiFileCompilationUnit variable in the *modelsim.ini* file..

- -mixedansiports

  Use this switch only when your design files contain a combination of ANSI and non-ANSI port declarations and task/function declarations. For example:

```
module top (input reg [7:0] a,
            output b);
   reg [7:0] b;
endmodule
```

- -mixedsvvh [b | s | v]

  (optional) Facilitates using SystemVerilog packages at the SystemVerilog-VHDL boundary of a mixed-language design. When you compile a SystemVerilog package with -mixedsvvh, the package can be included in a VHDL design as if it were defined in VHDL itself.

  b — treats all scalars/vectors in the package as VHDL bit/bit_vector

  s — treats all scalars/vectors in the package as VHDL std_logic/std_logic_vector

- -modelsimini <path/modelsim.ini>

  Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable. Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -nocovercells

  (optional) Disables code coverage of modules defined by `celldefine and `endcelldefine compiler directives, or compiled with the -v or -y arguments. Can be used to override the CoverCells compiler control variable in the *modelsim.ini*.

- -nocoverexcludedefault

  (optional) Allows Verilog/SV branch code coverage data collection for the default branch in case statements.

- -nocoverfec

  (optional) Disables focused expression coverage (FEC) for coverage collection. By default, FEC coverage statistics are disabled for collection. You can customize the default behavior with the CoverFEC variable in the *modelsim.ini* file.

- -nocoverrec

  (optional) Disables rapid expression coverage (REC) for coverage collection. By default, REC coverage statistics are enabled for collection. Disabling REC collection converts non-masking conditions in FEC tables to matching input patterns. You can customize the default behavior with the CoverREC variable in the *modelsim.ini* file.

- -nocovershort

  (optional) Disables short circuiting of expressions when coverage is enabled. Short circuiting is enabled by default. You can customize the default behavior with the CoverShortCircuit variable in the *modelsim.ini* file.

- -nocoverudp

  (optional) When UDP coverage has been enabled, it prevents UDP coverage statistics from being collected. By default, only FEC coverage statistics are enabled for collection. You can customize the default behavior with the CoverUDP variable in the *modelsim.ini* file.

- -nocreatelib

  (optional) Stops automatic creation of missing work libraries and reverts back to 10.3x and earlier version behavior. Overrides the CreateLib modelsim.ini variable.

- -nodbgsym

  Disables the generation of the symbols debugging database in the compiled library.

  The symbols debugging database is the .dbs file in the compiled library that provides information to the GUI allowing you to view detailed information about design objects at the source level. Two major GUI features that use this database include source window annotation and textual dataflow.

  You should only specify this switch if you know that anyone using the library will not require this information for design analysis purposes.

- -nodebug[=ports | =pli | =ports+pli]

  (optional) Hides, within the GUI and other parts of the tool, the internal data of all compiled design units.

  -nodebug — The switch, specified in this form, does not hide ports, due to the fact that the port information may be required for instantiation in a parent scope.

  The design units' source code, internal structure, registers, nets, and so on, will not display in the GUI.

  In addition, none of the hidden objects may be accessed through the Dataflow or Schematic window or with commands.

  This also means that you cannot set breakpoints or single step within this code. It is advised that you not compile with this switch until you are done debugging.

  Note that this is not a speed switch like the "nodebug" option on many other products.

  Use the vopt command to increase simulation speed.

-nodebug=ports — additionally hides the ports for the lower levels of your design; it should be used only to compile the lower levels of the design. If you hide the ports of the top level you will not be able to simulate the design.

Do not use the switch in this form when the parent is part of a vopt -pdu (black-box) flow or for mixed language designs, especially for Verilog modules to be instantiated inside VHDL.

-nodebug=pli — additionally prevents the use of pli functions to interrogate individual modules for information.

You should be aware that this form will leave a "nodebug" module untraversable by PLI.

-nodebug=ports+pli — you can combine the behavior of =ports and =pli in this manner.

This functionality encrypts entire files. The `protect compiler directive allows you to encrypt regions within a file.

- -noexcludeternary <design_unit>

  (optional) Disables the automatic exclusion of UCDB coverage data rows resulting from ternary expressions for the specified design unit. Normal operation for code coverage is to include rows corresponding to the case where two data inputs are the same, and the select input is a “don’t care”. To disable this automatic exclusion for the entire design, use “vsim -noexcludeternary” instead.

- -noForceUnsignedToVhdlInteger

  Prevents untyped Verilog parameters in mixed-language designs that are initialized with unsigned values between 2*31-1 and 2*32 from being converted to a VHDL generic. By default, untyped Verilog parameters that are initialized with unsigned values between 2*3 -1 and 2*32 are converted to VHDL INTEGER generics. Because VHDL INTEGER parameters are signed numbers, the Verilog values 2*31 -1 to 2*32 are converted to negative VHDL values in the range from -2*31 to -1 (the 2's complement value).

- -noincr

  (optional) Disables incremental compilation previously turned on with -incr argument. Default.

- -nologo

  (optional) Disables the startup banner.

- -nopsl

  (optional) Causes the compiler to ignore embedded PSL assertions and prevents parsing of any code within the PSL metacomment including any HDL code. By default, vlog parses any PSL assertion statements it finds in the specified files. Refer to “Simulating Assertions” for more information.

- -nosparse

  (optional) Turn off sparse memory processing for memories that would otherwise be implemented with sparse storage. Identifies which memories are considered “not sparse,”

which causes Questa SIM to override the rules for allocating storage for memory elements only when necessary. If you use -nosparse on a given memory, Questa SIM will simulate the memory normally. Refer to Sparse Memory Modeling for more information.

- +nospecify

  (optional) Disables specify path delays and timing checks.

- -note <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "note." Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- +notimingchecks

  (optional) Removes all timing check entries from the design as it is parsed.

  To disable checks on individual instances, use the tcheck_set command.

- -novopt

  (optional) Forces vlog to produce code if the VoptFlow variable is set to 1 (optimizations turned on) in the *modelsim.ini*. (VoptFlow = 1 is the default behavior.) Use this argument together with the vsim -novopt command to run the simulator without any optimizations. Use of this switch is not recommended except when using Power Aware simulation together with QuestaADMS. If this switch is used, the following warning is produced:

  ```
  ** Warning: (vopt-%d) All optimizations are turned off because the
  -novopt switch is in effect. This will cause your simulation to run
  very slowly. If you are using this switch to preserve visibility for
  Debug or PLI features, please see the User's Manual section on
  Preserving Object Visibility with vopt.
  ```

- -novtblfixup

  Causes virtual method calls in SystemVerilog class constructors to behave as they would in normal class methods, which prevents the type of a `this` reference from changing during construction.

  This overrides default behavior, where the type of a `this` reference is treated as if it is a handle to the type of the active `new()` method while a constructor is executing (which implies that virtual method calls resolve will not execute methods of an uninitialized class type).

- +nowarn<CODE>

  (optional) Disables warning messages in the category specified by <CODE>. Warnings that can be disabled include the <CODE> name in square brackets in the warning message. For example,

  ```
  ** Warning: test.v(15): [RDGN] - Redundant digits in numeric
  literal.
  ```

  This warning message can be disabled by specifying +nowarnRDGN.

- -nowarn <category_number>

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress. Multiple -nowarn switches are allowed. Warnings may be disabled for all compiles via the Main window **Compile > Compile Options** menu command or the *modelsim.ini* file (refer to modelsim.ini Variables).

  The warning message categories are described in Table 2-14:

Table 2-14. Warning Message Categories for vlog -nowarn

Category number	Description
11	PSL warnings
12	non-LRM compliance in order to match Cadence behavior
13	constructs that code coverage can't handle
15	SystemVerilog assertions using local variable

- +num_opt_cell_conds+<value>

  (optional) Restricts gate-level optimization capacity for accepting cells with I/O path and timing check conditions.

  value — integer between 32 and 1023, inclusive. where the default value is 1023.

- -optionset <optionset_name>

  (optional) Calls an optionset as defined in the *modelsim.ini file*. Refer to the section "Optionsets" for more information.

- -outf <filename>

  (optional) Specifies a file to which the final list of options is saved, after recursively expanding all -f, -file and -F files.

- -override_timescale[=]|[ ]<time_unit> / <time_precision>

  (optional) Specifies a timescale for all compiled design units. This timescale overrides all 'timescale directives and all declarations of timeunit and timeprecision. An equal sign (=) or whitespace is accepted between option and arguments.

  time_unit — unit of measurement for times and delays. This specification consists of one of three integers (1, 10, or 100) representing order of magnitude and one of six character strings representing units of measurement:

  {1 | 10 | 100} {s | ms | us | ns | ps | fs}

  For example, 10 ns.

  time_precision — unit of measurement for rounding delay values before being used in simulation. Allowable values are the same as for time_unit.

- -O0 | -O1 | -O4 | -O5

  (optional) Lower the optimization to a minimum with -O0 (capital oh zero). Use this to work around bugs, increase your debugging visibility on a specific cell, or when you want to place breakpoints on source lines that have been optimized out.

  Please refer to the section "Optimizing Designs with vopt" in the User's Manual for detailed information on using vopt to perform optimization.

  - (optional) Enable PE-level optimization with -O1.
  - (default) Enable standard SE optimizations with -O4.
  - (optional) Enable maximum optimization with -O5. The tool attempts to optimize loops and prevents variable assignments in situations where a variable is assigned but is not actually used.

- -pedanticerrors

  (optional) Enforces strict compliance of the IEEE Std 1800-2005. The following are some of the cases:

  - Covergroup bin size, value range, or transition specification must be constant.
  - Using "new" for queues is not legal. When strict compliance is not enforced, use of "new" creates a queue of the specified size where all elements are initialized to the default value of the queue element type.
  - Using underscore character (_) in sized, based literals is not legal. When you specify this argument, an error will occur for literals such as 2'b_01.
  - Omitting the grave accent mark (`) preceding the left brace ({) when writing structure literals is not legal. When you specify this argument, an error will occur for literals written without that mark.
  - Inserting the grave accent mark to precede quotation marks (`") that enclose string literals is not legal—only string literals within quotation marks (") are allowed. When you specify this argument, an error will occur for string literals using that mark.
  - Using class extern method prototypes with lifetime (automatic/static) designations produces a compliance error (instead of a warning).
  - Using "cover bool@clk" as a PSL statement.
  - Using realtime data types in SystemVerilog assertions.
  - Using an unsized constant in a concatenation if it is the leftmost value in the list.
  - Calling a virtual function in the constructor of the same class.
  - Using integers to define macro names.

  This argument also produces a report of mismatched 'else directives.

  You can produce a complete list by executing the command:

```
verror -kind vlog -pedanticerrors
```

- -permissive

  (optional) Allows messages in the LRM group of error messages to be downgraded to a warning. Allows reserved keywords 'config' and 'instance' to be used outside of unit and configuration scopes. Also allows named port connections on bit-select and part-select ports, though only when multiple bit-select or part-select ports of same name are not present in the port list.

  You can produce a complete list by executing the command:

  ```
  verror -kind vlog -permissive
  ```

- -permit_defunct_sv

  (optional) Allows using a selected set of constructs no longer supported by the SystemVerilog standard. Currently, the set supports only the use of the keyword “char.” This argument allows use of the keyword “char” to be interpreted as the SystemVerilog “byte” type.

- -primitiveaccess [=[-]<accesscodes>]

  (optional) Allow access of PLI handles to primitive instances.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

- -printinfilenames[=<filename>]

  Prints the path names of all source files opened (including “include” files) during the compile. Specifies whether each file is a Verilog or SystemVerilog file. To write these path names to a text file in the current directory, add =<filename> to this argument. If you use this argument again with the same filename, you overwrite the contents of the previous version of the file.

- -[w]prof=<filename>

  (optional; -wprof and -prof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- +protect[=<filename>]

  (optional) Enables `pragma protect directives for encrypting selected regions of your source code. Produces an encrypted output file saved in the work library and appends the letter ‘p’ to the extension, for example, *.vp*. To save an encrypted output file to the current directory, add =<filename> to this argument. If you specify a filename the output is the concatenation of all of the encrypted versions of the input source files.

  Any `include files will also be inserted into the output file when you add =<filename>. If you do not use =<filename>, all `include files will be encrypted into the work directory as individual files, not merged together into one file.

- -pslext

  (optional) Enables PSL LTL and OBE operators. These operators are disabled by default.

- -pslfile <filename>

  Identifies an external PSL assertion file to compile along with the Verilog source files. Refer to “Simulating Assertions” for more information.

- -quiet

  (optional) Disables 'Loading' messages.

- -R [<simargs>]

  Instructs the compiler to invoke vsim after compiling the design. The compiler automatically determines which top-level modules are to be simulated.

  When using -R option, the log files for vlog and vsim need to be specified separately. The file specified before -R will capture the output of the vlog compiler and the one provided after -R will capture the vsim output.

  For example, in the following vlog command, "log1.txt" will contain the vlog output and "log2.txt" will contain the vsim output.

  ```
  vlog -l log1.txt top.sv -R -c -do "run -all;quit" -l log2.txt
  ```

  The -R option is not a Verilog-XL option, but it is used by Questa SIM to combine the compile and simulate phases together as you may be used to doing with Verilog-XL. It is not recommended that you regularly use this option because you will incur the unnecessary overhead of compiling your design for each simulation run. Mainly, it is provided to ease the transition to Questa SIM.

- -randmetastable [=[-]<accesscodes>]

  (optional) Enable random insertion of 0 or 1 values (1’b0 or 1’b1) on the output of a library cell when a timing violation is encountered.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

r — Preserve read access on selected object.

w — Preserve write access on selected object.

- -refresh

  (optional) Regenerates a library image. By default, the work library is updated. To update a different library, use -work <library_name> with -refresh (for example, vlog -work <your_lib_name> -refresh). If a dependency checking error occurs which prevents the refresh, use the vlog -force_refresh argument. See vlog examples for more information. You may use a specific design name with -refresh to regenerate a library image for that design, but you may not use a file name.

- -s

  (optional) Instructs the compiler not to load the standard package. This argument should only be used when you are compiling the sv_std package.

- -scdpiheader <filename>

  (optional) Specifies the name of SystemC DPI function prototype header file automatically generated from the current compilation. The default filename is *sc_dpiheader.h* when no such switch is provided. Refer to section SystemC Procedural Interface to SystemVerilog for more detailed description.

- -sfcu

  Instructs the compiler to treat all files within a compilation command line as a separate compilation units. This is the default behavior and is the inverse of the behavior of -mfcu[=macro].

  This switch will override the MultiFileCompilationUnit variable if it is set to "1" in the *modelsim.ini* file.

- -skipprotected

  (optional) Ignores any ‘protected/‘endprotected region contained in a module.

- -skipprotectedmodule

  (optional) Prevents adding any module containing a ‘protected/‘endprotected region to the library.

- -skipsynthoffregion

  (optional) Ignore all constructs within synthesis_off or translate_off pragma regions.

- -smartdbgsym

  (optional) Reduces the size of design libraries by minimizing the amount of debugging symbol files generated at compile time.

  Code Coverage flows do not support this option and there are limitations to `macro support in refresh flows.

  Edit the SmartDbgSym variable in the *modelsim.ini* file to set a permanent default.

- -source

  (optional) Displays the associated line of source code before each error message that is generated during compilation. By default, only the error message is displayed.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of compiler statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the settings in the Stats *modelsim.ini* variable.

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

  time — (default) Display Start, End, and Elapsed times.

  Modes

  Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vlog -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vlog -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

  kb — Print performance statistics in Kb units with no auto-scaling.

  list — Display statistics in a Tcl list format when available.

  verbose — Display verbose statistics information when available.

Note

vlog -quiet disables all default or user-specified -stats features.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message you wish to suppress. You cannot suppress Fatal or Internal messages. Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -sv

  (optional) Enables SystemVerilog features and keywords. By default Questa SIM follows the IEEE Std 1364-2001 and ignores SystemVerilog keywords. If a source file has a ".sv" extension, Questa SIM will automatically parse SystemVerilog keywords.

- -svext=[+|-]<extension>[,[+|-]<extension>]…

  (optional) Enables SystemVerilog language extensions through a comma-separated list of arguments.

  [+ | -] — controls activation of the *extension*. Remember that arguments on the command line control the precedence; any settings to this switch will override your settings of the SvExtensions *modelsim.ini* variable.

  + — activates the *extension*.

  - — deactivates the *extension*.

  If you do not specify either a "+" or "-", the command assumes you are activating the specified *extension*.

  <extension> —

  acum — Specifies that the get(), try_get(), peek(), and try_peek() methods on an untyped mailbox will return successfully if the argument passed is assignment-compatible with the entry in the mailbox. The LRM-compliant behavior is to return successfully only if the argument and entry are of equivalent types.

  atpi — Use type names as port identifiers. Disabled when compiling with -pedanticerrors.

  ared — Allows use of array reduction methods on multi-dimensional unpacked arrays, without the need of using a 'with' clause. A multi-dimensional unpacked array will be treated as if it had a single dimension [0:total_number_of_elements-1].

  catx — Allow an assignment of a single unsized constant in a concat to be treated as an assignment of 'default:val'.

  daoa — Allows the passing a dynamic array as the actual argument of DPI open array output port. Without this option, a runtime error, similar to the following, is generated, which is compliant with LRM requirement.

```
# ** Fatal: (vsim-2211) A dynamic array cannot be passed as an
argument to the DPI import function 'impcall' because the formal 'o'
is an unsized output.
#    Time: 0 ns  Iteration: 0  Process: /top/#INITIAL#56 File:
dynarray.sv
# Fatal error in Module dynarray_sv_unit at dynarray.sv line 2
```

evdactor — enables early variable declaration assignments during class construction. The default behavior is to perform all superclass initialization before initializing any fields in a subclass.

evis — Supports the expansion of environment variables within `include path names. For example, if MYPATH exists in the environment then it will be expanded in the following:

```
`include "$MYPATH/inc.svh"
```

feci — Treat constant expressions in a foreach loop variable index as constant.

fin0 — Treats $finish() system call as $finish(0), which results in no diagnostic information being printed.

ias — Iterate on always @* evaluations until inputs settle. Typically, an always @* block is not sensitive to events generated by executing the block itself. This argument increases the sensitivity of the block so that it will re-trigger if any input has changed since the last iteration of the always block.

idcl — Allows passing of import DPI call locations as implicit scopes.

iddp — Ignore the DPI task disable protocol check.

pae — Automatically export all symbols imported and referenced in a package.

sccts — Process string concatenations converting the result to string type.

spsl — (default) Search for packages in source libraries specified with -y and +libext.

stop0 — Treats $stop and $stop() as $stop(0), which results in no diagnostic information being printed.

udm0 — Expands any undefined macro with the text “1'b0”.

uslt — (default) Promote unused design units found in source library files specified with the -y option to top-level design units.

vmctor — Allows virtual method calls in class constructor. The default is to treat them as non-virtual during construction.

Multiple extensions are specified as a comma separated list. For example:

vlog -svext=+feci,-uslt,pae

- -svfilesuffix=<extension>[,<extension>...] <filename>

  Allows specification of filename extensions for SystemVerilog files. Overrides the SVFileSuffixes variable in the *modelsim.ini* file for specified <filename>.

- -svinputport=net | var | relaxed

  (optional) Used in conjunction with -sv to determine the default data type assigned to an input port declaration.

  net — declares the port to be a net. This value enforces strict compliance to the Verilog LRM (IEEE Std 1364-2005), where the port declaration defaults to wire.

  var — declares the port to be a variable. This value enforces behavior from previous releases, where the port declaration defaults to variable.

relaxed — (default) declares the port to be a net only if the type is a 4-state scalar or 4-state single dimensional vector. Otherwise, the port is declared a variable.

- -svpkgcasesens

  (optional) Requires case-sensitive matching between SystemVerilog package import statements and package names.

- -sv05compat

  Used in conjunction with the -sv switch to ensure compatibility with the reserved keyword set of IEEE Std 1800-2005.

- -sv09compat

  Used in conjunction with the -sv switch to ensure compatibility with the reserved keyword set of IEEE Std 1800-2009.

- -sv12compat

  Used in conjunction with the -sv switch to ensure compatibility with the reserved keyword set of IEEE Std 1800-2012.

- -systfoverride [=[-]<accesscodes>]

  (optional) Allow user-specified overrides of built-in system tasks (such as $write or $display).

  NOTE: In order for specified overrides to take effect in certain design regions, you must also ensure that vopt -systfoverride is in effect. If it is not in effect, the system-provided system task will remain in effect, even if you have made an overridden version available with PLI.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

- -tbxhvllint <hvl_files>

  (optional. For use with Veloce emulator and TestBench-Xpress (TBX) verification accelerator) Prints a warning message for each #delay statement encountered during analysis. Refer to the *TBX User's Guide* for more detailed usage information.

- -timescale[=]|[ ]<time_units>/<time_precision>

  (optional) Specifies the default timescale for all design unit types (modules, interfaces, programs, packages, checkers, and so forth) not having an explicit timescale directive in effect during compilation.

  The format of the -timescale argument is the same as that of the `timescale directive. An equal sign (=) or whitespace is accepted between option and arguments in which case

<time_units / <time_precision must be enclosed in quotation marks ("). The format for *<time_units>* and *<time_precision>* is *<n><units>*. The value of *<n>* must be 1, 10, or 100. The value of *<units>* must be fs, ps, ns, us, ms, or s. In addition, the *<time_precision>* must be smaller than or equal to the *<time_units>*. Refer to "Simulator Resolution Limit (Verilog)" for more information.

- -togglecountlimit <int>

  (optional) Limits the toggle coverage count, <int>, for a toggle node. After the limit is reached, further activity on the node is ignored for toggle coverage. All possible transition edges must reach this count for the limit to take effect. For example, if you are collecting toggle data on 0->1 and 1->0 transitions, both transition counts must reach the limit. If you are collecting "full" data on 6 edge transitions, all 6 must reach the limit. Overrides the global value set by the ToggleCountLimit *modelsim.ini* variable.

- -toggleportsonly

  (optional) Enables only ports for inclusion in toggle coverage numbers; no internal signals are included in coverage metrics. If you want to see coverage of all ports in the design, use the "vopt +acc=p" command — but note that there could be a significant performance penalty due to the fact that inlining is turned off. You can selectively enable toggle coverage on specific ports by using the "vopt +acc=p+<selection>" command. Overrides any global value set by the TogglePortsOnly *modelsim.ini* variable.

- -togglewidthlimit <int>

  (optional) Sets the maximum width of signals, <int>, that are automatically added to toggle coverage with the -cover t argument. Can be set on design unit basis. Overrides the global value of the ToggleWidthLimit *modelsim.ini* variable.

- +typdelays

  Selects typical delays from the "min:typ:max" expressions. Default. If preferred, you can defer delay selection until simulation time by specifying the same option to the simulator.

- -u

  (optional) Converts regular Verilog identifiers to uppercase. Allows case insensitivity for module names.

- -v <library_file>

  (optional) Specifies a source library file containing module and UDP definitions. Refer to "Verilog-XL Compatible Compiler Arguments" for more information.

  After all explicit filenames on the vlog command line have been processed, the compiler uses the -v option to find and compile any modules that were referenced but not yet defined. Modules and UDPs within the file are compiled only if they match previously unresolved references. Multiple -v options are allowed. See additional discussion in the examples.

- -version

  (optional) Returns the version of the compiler as used by the licensing tools.

- -vlog01compat

  (default) Ensures compatibility with rules of IEEE Std 1364-2001.

- -vlog95compat

  (optional) Disables Verilog 2001 keywords, which ensures that code that was valid according to the 1364-1995 spec can still be compiled. By default Questa SIM follows the rules of IEEE Std 1364-2001. Some requirements in 1364-2001 conflict with requirements in 1364-1995. Edit the vlog95compat variable in the *modelsim.ini* file to set a permanent default.

- -vopt

  (optional) Notifies vlog that the vopt command will be run. As a result, vlog does not produce code. This saves an unnecessary code generation step.

  This argument is needed only if you have set the VoptFlow variable to 0 in the *modelsim.ini* file. If it is set to 1 (default operation) vlog skips the code generation step automatically.

- -vmake

  Generates a complete record of all command line data and files accessed during the compile of a design. This data is then used by the vmake command to generate a comprehensive makefile for recompiling the design library. By default, vcom stores compile data needed for the -refresh switch and ignores compile data not needed for -refresh. The -vmake switch forces inclusion of all file dependencies and command line data accessed during a compile, whether they contribute data to the initial compile or not. Executing this switch can increase compile time in addition to increasing the accuracy of the compile. See the vmake command for more information.

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information.

- -warning error

  (optional) Reports all warnings as errors.

- -warnrbw

  (optional) Displays a warning when a variable is read before written in an always @* block.

- -work <library_name>

  (optional) Specifies a logical name or pathname of a library that is to be mapped to the logical library work. By default, the compiled design units are added to the work library. The specified pathname overrides the pathname specified for work in the project file.

- -writetoplevels <fileName>

  (optional) Records the names of all top level module names in a specified file. Also records any compilation unit name specified with -cuname. May only be specified when compiling the top level modules.

<fileName> — Required. Specifies the name of the file where module names are to be recorded.

- -y <library_directory>

  (optional) Specifies a source library directory containing definitions for modules, packages, interfaces, and user-defined primitives (UDPs). Usually, this is a directory of source files that you want to scan if the compiled versions do not already exist in a library. Refer to "Verilog-XL Compatible Compiler Arguments" for more information.

  After all explicit filenames on the vlog command line have been processed, the compiler uses the -y option to find and compile any modules that were referenced but not yet defined. Files within this directory are compiled only if the file names match the names of previously unresolved references. Multiple -y options are allowed. You will need to specify a file suffix by using -y in conjunction with the +libext+<suffix> option. See additional discussion in the examples.

Note

Any -y arguments that follow a -refresh argument on a vlog command line are ignored. Any -y arguments that come before the -refresh argument on a vlog command line are processed.

- <filename>

  Specifies the name of the Verilog source code file to compile. One filename is required. Multiple filenames can be entered separated by spaces. Wildcards can be used.

Examples

- Compile the Verilog source code contained in the file *example.vlg*.

```
vlog example.vlg
```

- Hide the internal data of *example.v*. Models compiled with -nodebug cannot use any of the Questa SIM debugging features; any subsequent user will not be able to see into the model.

```
vlog -nodebug example.v
```

- The first line compiles and hides the internal data, plus the ports, of the lower-level design units, *level3.v* and *level2.v*. The second line compiles the top-level unit, *top.v*, without hiding the ports. It is important to compile the top level without =ports because top-level ports must be visible for simulation.

```
vlog -nodebug=ports level3.v level2.v
vlog -nodebug top.v
```

  The first command hides the internal data, and ports of the design units, *level3.v* and *level2.v*. In addition it prevents the use of PLI functions to interrogate the compiled modules for information (either =ports+pli or =pli+ports works fine for this command).

The second line compiles the top-level unit without hiding the ports but restricts the use of PLI functions as well.

- Note that the =pli switch may be used at any level of the design but =ports should only be used on lower levels since you can't simulate without visible top-level ports.

```
vlog -nodebug=ports+pli level3.v level2.v
vlog -nodebug=pli top.v
```

- After compiling *top.v*, vlog will scan the file *und1* for modules or primitives referenced but undefined in *top.v*. Only referenced definitions will be compiled.

```
vlog top.v -v und1
```

- After compiling *top.v*, vlog will scan the *vlog_lib* library for files with modules with the same name as primitives referenced, but undefined in *top.v*. The use of +libext+.v+.u implies filenames with a *.v* or *.u* suffix (any combination of suffixes may be used). Only referenced definitions will be compiled.

```
vlog top.v +libext+.v+.u -y vlog_lib
```

The -work option specifies mylib as the library to regenerate. -refresh rebuilds the library image without using source code, allowing models delivered as compiled libraries without source code to be rebuilt for a specific release of Questa SIM.

- If your library contains VHDL design units, be sure to regenerate the library with the vcom command using the -refresh option as well. Refer to "Regenerating Your Design Libraries" for more information.

```
vlog -work mylib -refresh
```

- The -incr option determines whether or not the module source or compile options have changed as *module1.v* is parsed. If no change is found, the code generation phase is skipped. Differences in compile options are determined by comparing the compiler options stored in the _info file with the compiler options given. They must match exactly.

```
vlog module1.v -u -O0 -incr
```

- The -timescale option specifies the default timescale for *module1.v*, which did not have an explicit timescale directive in effect during compilation. Quotes (" ") are necessary because the argument contains white spaces.

```
vlog module1.v -timescale "1 ns / 1 ps"
```

- The -fsmmultitrans option enables detection and reporting of multi-state transitions when used with the +cover f argument.

```
vlog +cover=f -fsmmultitrans
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vlog -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all default settings and then enables the perf option.

```
vlog -stats=time,cmd,msg -stats=none,perf
```

vmake

Requirement: This command does not work with the default format of the output from the vlib command, therefore you must add the -type directory argument to the vlib command line.

The vmake utility allows you to use a MAKE program to maintain individual libraries. You run vmake on a compiled design library. This utility operates on multiple source files per design unit; it supports Verilog include files as well as Verilog and VHDL PSL vunit files.

Note

If a design is spread across multiple libraries, then each library must have its own makefile and you must build each one separately.

By default, the output of vmake is sent to stdout—however, you can send the output to a makefile by using the shell redirect operator (>) along with the name of the file. You can then run the makefile with a version of MAKE (not supplied with Questa SIM) to reconstruct the library. *This command must be invoked from either the system prompt.*

A MAKE program is included with Microsoft Visual C/C++, as well as many other program development environments.

After running the vmake utility, MAKE recompiles only the design units (and their dependencies) that have changed. You run vmake only once; then you can simply run MAKE to rebuild your design. If you add new design units or delete old ones, you should re-run vmake to generate a new makefile.

The vmake utility ignores library objects compiled with -nodebug.

Also, the vmake utility is not supported for use with SystemC.

This command provides additional information with the -help switch.

Syntax

vmake [-cygdrive] [-du <design_unit_name> ...] [-f <filename>] [-fullsrcpath] [-ignore] [<library_name>] [-modelsimini <path/modelsim.ini>]

Arguments

- -cygdrive

  (optional) Generates a makefile that uses a path specified with Linux pathname conventions. Use this argument if you are using cygwin v3.81 or later (which no longer supports Windows conventions for drive and pathname).

- -du <design_unit_name>

  (optional) Specifies that a vmake file will be generated only for the specified design unit. You can specify this argument any number of times for a single vmake command.

- -f <filename>

  (optional) Specifies a file to read command line arguments from.

Refer to the section "Argument Files" for more information

- -fullsrcpath

  (optional) Produces complete source file paths within generated makefiles. By default, source file paths are relative to the directory in which compilations originally occurred. Use this argument to copy and evaluate generated makefiles within directories that are different from where compilations originally occurred.

- -ignore

  (optional) Omits a make rule for the named primary design unit and its secondary design units.

- <library_name>

  (optional) Specifies the library name; if none is specified, then work is assumed.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified by the MODELSIM environment variable. Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems, the path separator should be a forward slash (/).

Examples

- To produce a makefile for the work library:

  ```
  vmake >mylib.mak
  ```

- To run vmake on libraries other than work:

  ```
  vmake mylib >mylib.mak
  ```

- To rebuild mylib, specify its makefile when you run MAKE:

  ```
  make -f mylib.mak
  ```

vmap

The vmap command defines a mapping between a logical library name and a directory by modifying the *modelsim.ini* file.

With no arguments, vmap reads the appropriate *modelsim.ini* file(s) and prints to the transcript the current logical library to physical directory mappings.

This command provides additional information with the -help switch.

Syntax

vmap [-c | -del <logical_name> ... | <logical_name> [<path>]]
[-modelsimini <path/modelsim.ini>]

Arguments

- -c

  (optional) Copies the default *modelsim.ini* file from the Questa SIM installation directory to the current directory.

  This argument is intended only for making a copy of the default *modelsim.ini* file to the current directory. Do not use it while making your library mappings or the mappings may end up in the incorrect copy of the *modelsim.ini*.

- -del <logical_name> ...

  (optional) Deletes the mapping specified by <logical_name> from the current project file. You can specify multiple logical name arguments to the -del switch to delete multiple library mappings.

- <logical_name> [<path>]

  (optional) Maps a logical library name to the specified physical library.

  If you do not specify <path> the command returns the current mapping for <logical_name>.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable. Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

Examples

- Map two logical libraries to the physical library “work”:

```
vlib work
vmap library1 work
vmap library2 work
```

- Display information about the logical library “library1”:

```
vmap library1
```

- Delete the logical library mappings:

```
vmap -del library1 library2
```

vopt

The vopt command performs global optimizations on designs after they have been compiled with vcom or vlog. For detailed usage information on optimization, refer to the chapter titled “Optimizing Designs with vopt” in the User’s Manual.

Note

The default optimization behavior of vopt may differ from what is documented in this section if you are using any *modelsim.ini* file other than the one you installed with Questa SIM. See “Optimizing Designs with vopt” and "VoptFlow" in the User’s Manual for information on possible optimization settings.

The vopt command produces an optimized version of your design in the working directory. You provide a name for this optimized version using the -o argument. You can then invoke vsim directly on that new design unit name.

The vopt command makes use of the PathSeparator variable from the *modelsim.ini* file, which allows you to specify the character that Questa SIM recognizes as a path separator. By default, vopt expects you to use the forward-slash (/), but you could change this to a period (.) for ease of use with Verilog designs. For example:

```
vopt top -o opttop +acc=rn+.top.middle.bottom -G.top.myparam=4
```

The default operation of vopt -o <name> is incremental compilation—that is, Questa SIM reuses elements of the design that have not changed. This improves performance when using vopt on a design that has been minimally modified.

In the course of optimizing a design, vopt will remove objects that are deemed unnecessary for simulation. For example, line numbers are removed, processes are merged, nets and registers may be removed, and so on. If you need visibility into your design for debugging purposes, use the vopt accessibility arguments (listed in Using vopt for Access Control for Visibility During Optimization) to conditionally enable visibility for parts of your design.

All arguments to the vopt command are case sensitive: -WORK and -work are not equivalent.

Note

Many of the vopt arguments listed below may also be specified at compile time (for example, -O5). These "in-common" arguments are processed by vopt automatically (for example, if you specify them at compile time, vopt will use them automatically during optimization). The first instance of an in-common argument takes priority.

For example, if you specify -O5 to vlog or vcom, and then specify -O1 to vopt, -O5 takes precedence.

There is one exception to this rule: vopt will "OR" any +cover arguments to vlog or vcom (vlog +cover=bc) with any +cover arguments to vopt (vopt +cover=est).

This command provides additional information with the -help or -h argument.

Syntax

vopt [options] <design_unit> -o <name>

[options]:

[-32 | -64]

[{+acc | +noacc}[=<spec>][+<selection>[+<recurse_level>|.]]]

Note

Use of the +acc argument with vopt is not recommended for the following reasons:

- It may reduce simulation speed.
- The +acc argument and many of its values have been replaced by a collection of individual arguments listed in Using vopt for Access Control for Visibility During Optimization. The +acc argument is supported but deprecated.

[-cellreport <filename>] [+check<enum>] [-compat]
[-constimmedassert | -noconstimmedassert] [-cppinstall <[gcc|g++] version>]
[-cpppath <filename>] [{+cover | +nocover}[=<spec>]
[+<selection>[+<recurse_level>|.]]]
[-coverclkoptbuiltins | -nocoverclkbuiltins] [-coverdeglitch {“<n> <time_unit>”}]
[-coverenhanced] [-coverexcludedefault | -nocoverexcludedefault]
[-coverfec | -nocoverfec] [-coverrec | -nocoverrec] [-coveropt <opt_level>]
[-coverreportcancelled] [-coverudp | -nocoverudp]

[-debugdb] [-debugCellOpt] [-deferSubpgmCheck]
[-deglitchalways | -nodeglitchalways] [+delay_mode_distributed]
[+delay_mode_path] [+delay_mode_unit] [+delay_mode_zero] [-dpiforceheader]
[-dpiheader <filename>]

[-enablescstdout] [-error <msg_number>[,<msg_number>,...]]
[-extendedtogglemode 1|2|3

[(-F | -file | -f) <filename>] [-fatal <msg_number>[,<msg_number>,…]]
[-fecudpeffort {1 | 2 | 3}] [-floatgenerics[+<selection>[+<recurse_level>|.]]]
[-floatparameters[[+<selection>[+<recurse_level>|.]] | [+<leafparameter>]]]
[-fsmimplicittrans] [-fsmmultitrans] [-fsmresettrans | -nofsmresettrans]
[-fsmsingle | -nofsmsingle] [-fsmverbose [b | t | w]]

[-g <Name>=<Value> …] [-G <Name>=<Value> …] [-gblso
<shared_obj>[,<shared_obj> …]]

[-hazards]

[-incr | -noincr] [+initmem[=<spec>][+{0 | 1 | X | Z}][+<selection>[+<recurse_level>|.]]
| +noinitmem[+<selection>[+<recurse_level>|.]]]
[-initoutcompositeparam | -noinitoutcompositeparam]
([+initmem[=<spec>][+{0 | 1 | X | Z}][+<selection>[+<recurse_level>|.]] |

+noinitmem[+<selection>[+<recurse_level>|.]]])
(+initreg[=<spec>][+{0 | 1 | X | Z}][+<selection>[+<recurse_level>|.]] |
+noinitreg[+<selection>[+<recurse_level>|.]])

[-j {0 | 1| <n>}]

[-keep_delta]

[-libertyfiles=<filename> [,<filename>,<filename>,...]] [-L <libname>]
[-Lf <libname>] [-logfile <filename> | -l <filename>] [-lowercasepslpragma]
[-lrmclassinit]

[+maxdelays] [+mindelays] [-mti_trace_vlog_calls]

[-mixedansiports] [-modelsimini <path/modelsim.ini>]

[-nocovershort] [-nocreatelib] [-nodebug[=ports | =pli | =ports+pli]] [-noexcludeternary]
[-fsmresettrans | -nofsmresettrans] [-nofsmxassign] [+nolibcell] [-nologo]
[-noscelab] [-noscmainscopename] [+nosparse[+<selection>[+<recurse_level>|.]]]
[+nospecify] [-note <msg_number>[,<msg_number>,...]]
[+notimingchecks] [+nowarn<CODE>] [-nowarn <category_number>]
[+num_opt_cell_conds+<value>]

[-O0 | -O1 | -O4 | -O5] [-ocf <filename>] [-optionset <optionset_name>]

[-pdu] [-pdusavehierrefs[=<filename>]] [-pduspec{[+<instancePathName>] |
[+[<libName>.]<moduleName>]}[+facc=<fileName>]]
[-permissive] [-[w]prof=<filename>] [-proftick=<integer>]
[-pslext] [-pslfile_vh <filename>] [-pslfile_vl <filename>]

[-quiet]

[-sclib <library>]
[-sdfmin | -sdftyp | -sdfmax[@<delayScale>] [<instance>=]<sdf_filename>]
[-sdfmaxerrors <n>] [-showsubprograms | -noshowsubprograms] [-source]
[+staticchecks[=<args>]] [-staticchecksfile <filename>] [-staticchecksmdvhdl]
[-stats [=[+ | -]<feature>[,[+ | -]<mode>]]
[-suppress <msg_number>[,<msg_number>,...]]
[-svext=[+|-]<extension>[,[+|-]<extension>]…]

[-tab <tabfile>]
[-tbxhierrefs[[+<instancePathName>] | [+[<libName>.]<moduleName>]]
[+f=<fileName>]]
[-timescale[=]|[ ]<time_units>/<time_precision>]
[-togglecountlimit <int>] [-toggleportsonly] [-togglewidthlimit <int>] [+typdelays]

[-undefsyms={<args>}]

[-version]

[-warning <msg_number>[,<msg_number>,...]] [-warning error] [-work
<library_name>]

[-xprop[,mode=<modecode>][,report=<severitylevel>][,object=<objectselection>]]

Using vopt for Access Control for Visibility During Optimization

The following optional arguments are replacements for the +acc argument and its values. Most of these arguments can be negated by adding 'no' prefix to the (see Table 2-15). Note that the +acc argument is still supported for backward compatibility, but its use is deprecated in favor of the vopt visibility arguments with the following syntax:

vopt [-[no]assertdebug [=[-]<accesscodes>][+<objectselection>]]
[-[no]bitscalars [=[-]<accesscodes>][+<objectselection>]] [-[no]cellaccess [=[-]<accesscodes>][+<objectselection>]] [-[no]fsmdebug [=[-]<accesscodes>][+<objectselection>]]
[-floatparameters[[+<selection>[+<recurse_level>|.]] | [+<leafparameter>]]]
[-[no]linedebug [=[-]<accesscodes>][+<objectselection>]]
[-nosparse] [-[no]primitiveaccess [=[-]<accesscodes>][+<objectselection>]] [-[no]randmetastable [=[-]<accesscodes>][+<objectselection>]] [-[no]systfoverride [=[-]<accesscodes>][+<objectselection>]]
<design_unit> -o <name>

By default, these arguments affect all design units in the design. Modifiers to these arguments help maximize simulation performance while retaining access to objects of interest, which might otherwise be optimized away.

You can target a specific design unit or instance with +<selection> to limit the extent of one or more arguments to only that design unit or instance using following syntax:

- [<library>.]<entity>[(<architecture>)] (for VHDL design units)
- [<library>.]<design_unit> (for Verilog design units)
- <inst_name> (for Verilog/VHDL design regions and objects)
- [<library>.]<design_unit>/<inst_path> (for Verilog/VHDL instances)

When [<library>.]<design_unit> is specified, all instances of that design unit from the specified <library> are affected by the argument. If <library> is not specified, all instances of that design unit are considered. In VHDL, you may further qualify the selection using <architecture>.

When +<inst_name> is specified, only objects in that part of the design are affected.

If a trailing period (.) is present, the argument is also applied recursively downwards. Or, you can also use <recurse_level> to apply this argument to a specified number of levels under this scope, where recurse_level is any integer from 0 to 128 (a value of 0 specifies full recursion).

Negated arguments (with "no" prefix) allow objects to be optimized, and they can also take <selection> and <recurse_level> modifiers (refer to Examples).

Table 2-15 lists the arguments for access visibility with the corresponding arguments that they are replacing.

Table 2-15. vopt Arguments for Access Visibility Being Replaced

Previously Supported Argument[1]	Replacement Argument
+acc=a	-assertdebug
+acc=b	-bitscalar
+acc=c	-cellaccess
+acc=f	-fsmdebug
+floatparameters	-floatparameters
+floatgenerics	-floatgenerics
+acc=l	-linedebug
+nosparse	-nosparse
+acc=s	-systfoverride
+acc=u	-primitiveaccess
+acc=x	-randmetastable[2]
+noacc=a	-noassertdebug
+noacc=b	-nobitscalar
+noacc=c	-nocellaccess
+noacc=f	-nofsmdebug
+noacc=l	-nolinedebug
+noacc=s	-nosystfoverride
+noacc=u	-noprimitiveaccess
+noacc=x	-norandmetastable

1. The following values of +acc and +noacc are also deprecated, but they do not have replacement arguments and are still supported for backward compatibility:
+acc=m, +acc=n, +acc=p, +acc=r, +acc=t, +acc=v
+noacc=m, +noacc=n, +anocc=p, +noacc=r, +noacc=t, +noacc=v

2. Verilog cells only.

Using vopt for Power Aware Simulation

The following optional arguments are specific to Power Aware Simulation.

vopt <design_unit> -o <name> [-pa_behavlogfile <filename>]
[-pa_checks=<check_value>] [-pa_checksoption=<value>] [-pa_connectpgpin i | a | e | c]
[-pa_corrupt={real | integer} | -pa_nocorrupt={real | integer}]
[-pa_defertop] [-pa_disable=<value>[+<value>]…]
[-pa_dumplibertydb=<database_pathname>] [-pa_dumpupf <filename>]
[-pa_enable= <value>[+<value>]…] [-pa_excludefile <filename>]
[-pa_genrpt=[{nv | v} [{us | ud | u}] [b] [pa] [de] [cell] [conn]] [-pa_gls]
[-pa_hiersep <alphanum_character>] [-pa_intcrptval0]
[-pa_lib <library_pathname>] [-pa_libertyrefresh=<database_pathname>]
[-pa_libertyupdate] [-pa_lsthreshold <real>]
[-pa_loadlibertydb=<database_pathname>] [-pa_modeltype=2] [-pa_noinpcorr]
[-pa_nopartialontrans] [-pa_pgoutsynchchk] [-pa_prefix <hier_path>]
[-pa_replacetop <string>] [-pa_reportdir <pathname>] [-pa_reportfile <filename>]
[-pa_rslvnetcheck] [-pa_rtlinfo] [-pa_tclfile <filename>] [-pa_top=<pathname>]
[-pa_upf <filename>]
[-pa_upfextensions=<extension>[+<extension>]] [-pa_upfoverloadsourcecmd] [-pa_upfsyntaxchecks]
[-pa_upfversion=[1.0 | 2.0 | 2.1]]

Using vopt for Visualizer Debugging Environment Simulation

The following optional arguments are specific to Visualizer Debugging Environment simulation.

vopt [<vopt_args>] [-debug[,viseffort=full | high | low]] -designfile <filename>

Arguments

- -32 | -64

  Specifies whether vopt uses the 32- or 64-bit executable, where -32 is the default.

  These options apply only to the supported Linux operating systems.

  These options override the MTI_VCO_MODE environment variable, which applies only to executables used from the *<install_dir>/bin/* directory. Therefore, these options are ignored if you run vopt from an *<install_dir>/<platform>/* directory.

  You can only specify these options on the command line, therefore they are not recognized as part of a file used with the -f switch.

- {+acc | +noacc}[=<spec>][+<selection>[+<recurse_level>|.]]

  (optional) Enable PLI and debug command access to objects indicated by <spec> when optimizing a design. Use +noacc to disable access to objects indicated by <spec> at the completion of optimization. Refer to the section "Preserving Object Visibility for Debugging Purposes" in the User's Manual for more information.

Note

Using the +acc or +noacc arguments may reduce simulation speed. Refer to Using vopt for Access Control for Visibility During Optimization for recommendations for using these arguments.

=<spec> — optionally, one or more of the following characters. If <spec> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified entity.

a — Preserve PSL and SystemVerilog assertion and functional coverage data. Enables pass count logging in the Transcript window and assertion viewing in the Wave window. Also enables the complete functionality of vsim -assertdebug.
If you do not specify +acc=a, the tool only transcripts assertion failure messages and reports only failure counts in the assertion browser.
Note that if a PSL or SystemVerilog construct is being driven by a port signal, vopt may replace that signal name with its higher-level driver. So in this case, if you prefer the local port name, the +acc "p" option should also be specified (for example, +acc=ap).

b — Enable access to bits of vector nets. This is necessary for PLI applications that require handles to individual bits of vector nets. Also, some user interface commands require this access if you need to operate on net bits.

c — Enable access to library cells. By default any Verilog module containing a non-empty specify block may be optimized, and debug and PLI access may be limited. This option keeps module cell visibility.

f — Enable access to finite state machines.

l — Enable access to line number directives and process names.

m — Preserve the visibility of module, program, and interface instances.

n — Enable access to nets.

p — Enable access to ports. This disables the module inlining optimization, and is necessary only if you have PLI applications that require access to port handles.

q — Enable access to VHDL variables and generics.

r — Enable access to registers (including memories, integer, time, and real types).

s — Enable access to system tasks.

t — Enable access to tasks and functions.

u — Enable access to primitive instances.

v — Enable access to variables, constants, and aliases in processes (VHDL design units only) that would otherwise be merged due to optimizations. Disables an optimization that automatically converts variables to constants.

x — Allow metastability simulation of select Verilog cells.

<n> — An integer between 0 and 128 that specifies a recursion level downwards from the current level of the specified entity. A value of 0 applies full recursion. This is overridden if you specify the optional period (.) after <selection>.

+<selection>[+<recurse_level>|.] — enables access for specific Verilog design objects and/or regions.

<selection> — Can take one of the following forms:

1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

<library> — Any Verilog, SystemVerilog, or VHDL library.

<primary> — Specifies the design unit name.

<secondary> — Specifies VHDL architecture.

2. <instance_path> — A path to an instance.

3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

Note

Glob wildcards ("*" and "?") are accepted only for the specification of design unit(s).

<recurse_level> — Specifies the number of recursion levels that selection will occur downward from the current level of the specified module or instance. This can be any integer between 0 and 128, where a value of 0 applies full recursion.

. (period) — Indicates that selection occurs recursively downward from the specified module or instance.

You can specify multiple selections in a plus (+)-separated list. For example:

```
+acc=rn+top1+top2
```

Ensure that you do not put a space between any <spec> arguments and the +<selection> argument.

- All modules — do not include the *selection* argument.
- All modules matching a wildcard — use a "*" or "?" in the design unit name.

  ```
  +acc=rn+m*
  ```

  This gives access to all modules whose names begin with "m".
- Single module — include only a module name as the *selection* argument.

  ```
  +acc=rn+top
  ```
- Single module and all instances below — append a period to the module name.

  ```
  +acc=rn+top.
  ```
- Specific instance — include a leading slash and the instance path.

```
+acc=rn+/top/mid1/bottom/myReg
```

This instance can be a scope name as well, such as a generate block, named block at the top level of the design unit, or a program, interface, or module instance.

- Specific instance and all instances below — include a leading slash and the instance path and append a period to the path.

```
+acc=rn+/top/mid1/bottom/myReg.
```

- Preserve nets, ports, and registers for three levels downward from top.netlist1 — access specifiers are followed by the number of levels and the selection specification.

```
+acc=npr3+top.netlist1
```

- Specific instance within a submodule — do not include a leading slash with the instance path.

```
+acc=rn+mid/bottom/myReg
```

This form will find all instances of the module *mid* and apply +acc=rn to all occurrences of *bottom/myReg*. Therefore, if you have two instances of *mid* (*/top/mid1* and */top/mid2*), +acc=rn applies to */top/mid1/bottom/myReg* and */top/mid2/bottom/myReg*.

- -[no]assertdebug [=[-]<accesscodes>][+<objectselection>]

(optional) Enable or disable debugging SVA/PSL objects when used with vsim -assertdebug. By default, -assertdebug affects all design units.

=<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

c — Preserve connectivity access on selected object.

r — Preserve read access on selected object.

w — Preserve write access on selected object.

<objectselection> — Can take one of the following forms:

1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

<library> — Any Verilog, SystemVerilog, or VHDL library.

<primary> — Specifies the design unit name.

<secondary> — Specifies VHDL architecture.

2. <instance_path> — A path to an instance.

3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

- -[no]bitscalars [=[-]<accesscodes>][+<objectselection>]

  (optional) Enable or disable access to net bits.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

  <objectselection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

  <library> — Any Verilog, SystemVerilog, or VHDL library.

  <primary> — Specifies the design unit name.

  <secondary> — Specifies VHDL architecture.

  2. <instance_path> — A path to an instance.

  3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

- -[no]cellaccess [=[-]<accesscodes>][+<objectselection>]

  (optional) Enable or disable access to cell internal objects.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

  <objectselection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

  <library> — Any Verilog, SystemVerilog, or VHDL library.

  <primary> — Specifies the design unit name.

  <secondary> — Specifies VHDL architecture.

  2. <instance_path> — A path to an instance.

  3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

- -cellreport <filename>

  This option writes to the specified output file, a list of Verilog modules which qualified for and received gate-level cell optimizations. Optimized cells in a Preoptimized Design Unit (black-boxed region) are not reported. To determine the cells optimized in a PDU the "-cellreport" switch must be used on the vopt command line used in the PDU step.

- +check<enum>

  (optional) Inhibits various modes of cell-level optimization checking, where <enum> represents one of the modes listed below.

  ALL — Enables all +check modes described below.

  AWA — Prevents an Always block from driving another Always block or a sequential UDP.

  CLUP — Disables optimization of cells containing connectivity loops.

  DELAY — When used in conjunction with vopt +delay_mode_path (see below), disables optimization of Verilog modules with distributed delays and no path delays in favor of module inlining.

  DNET — Prevents both the port and the delayed port (created for negative setup/hold) to be used in the functional section of the cell.

  INTRI — Disallows inputs of type tri1 or tri0 in cell optimizations.

  IPDOP — Disallows procedural code where an input directly drives an output.

  NWOT — Disallows a notifier register to be written outside of a timing check.

  OPRD — Disallows an output port to be read internally by the cell.

  SUDP — Disallows a sequential UDP to drive another sequential UDP.

- -compat

  (optional) Disables optimizations that result in different event ordering than Verilog-XL.

  Questa SIM Verilog generally duplicates Verilog-XL event ordering, but there are cases where it is inefficient to do so. Using this option does not help you find event order dependencies, but it allows you to ignore them. Keep in mind that this option does not account for all event order discrepancies, and that using this option may degrade performance. Refer to "Event Ordering in Verilog Designs" for additional information.

- -constimmedassert

  (optional) Displays immediate assertions with constant expressions in the GUI, in reports, and in the UCDB. By default, immediate assertions with constant expressions are displayed in the GUI, in reports, and in the UCDB. Use this switch only if the -noconstimmedassert switch has been used previously, or if the ShowConstantImmediateAsserts variable in the vopt section of the *modelsim.ini* file is set to 0 (off).

- -cppinstall <[gcc|g++] version>

  (optional) Specifies the version of the desired GNU compiler supported and distributed by Mentor Graphics.

<[gcc|g++] version> — The version number of the GNU compiler to use. Use the same compiler version as specified on the sccom command line. For example:

```
sccom -cppinstall 4.5.0
```

- -cpppath <filename>

  (optional) Specifies the location of a g++ executable other than the default g++ compiler installed with Questa SIM. Overrides the CppPath variable in the *modelsim.ini* file.

- {+cover | +nocover}[=<spec>] [+<selection>[+<recurse_level>|.]]

  (optional) Enables/disables various coverage statistics collection on specified areas of the design.

  The vopt +cover/+nocover argument accepts the same +<selection> syntax that +acc accepts, and can be used to specify design units, instances, and recursive control with a trailing '.' character. When you recompile an inlined module with different +cover/+nocover settings than a previous compile, you will be forcing a recompile of its inline parent modules on your subsequent optimization/compile.

  <spec> — one or more of the following characters:

  b — Collect branch statistics.

  c — Collect condition statistics. Collects only FEC statistics, unless -coverudp is specified.

  e — Collect expression statistics, Collects only FEC statistics, unless -coverudp is specified.

  s — Collect statement statistics.

  t — Collect toggle statistics. Overridden if ‘x’ is specified elsewhere.

  x — Collect extended toggle statistics (Refer to “Toggle Coverage” for details). This takes precedence, if ‘t’ is specified elsewhere.

  f — Collect Finite State Machine statistics.

  <n> — A natural number specifying a recursive level downwards from the specified entity. This is overridden if you specify the optional period (.) after <selection>.

  +<selection>[+<recurse_level>|.] — enables access for specific Verilog design objects and/or regions.

  <selection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

  <library> — Any Verilog, SystemVerilog or VHDL library.

  <primary> — Specifies the design unit name.

  <secondary> — Specifies VHDL architecture.

  2. <instance_path> — A path to an instance.

  3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

Note
Glob wildcards (“*” and “?”) are accepted only for the specification of design unit(s).

<recurse_level> — Specifies the number of recursion levels that selection will occur downward from the current level of the specified module or instance. This can be any integer between 0 and 128, where a value of 0 applies full recursion.

. (period) — Indicates that selection occurs recursively downward from the specified module or instance.

- All modules — do not include the *selection* argument.
- Single module — include only a module name as the *selection* argument.

```
+cover=bce+top
```

- Single module and all instances below — append a period to the module name.

```
+cover=bce+top.
```

- Single module and three levels of hierarchy below — add “3” to the <spec> and remove the period (.).

```
+cover=bce3+top
```

- Specific instance — include a leading slash and the instance path

```
+cover=bce+/top/mid1/bottom/myReg
```

- Specific instance and all instances below — include a leading slash and the instance path and append a period to the path.

```
+cover=bce+/top/mid1/bottom/myReg.
```

See the -coveropt <opt_level> argument to override the default level of optimization for coverage for a particular compilation run. Refer to “Rules for Applying Coverage with cover and nocover” for information on how potentially conflicting commands are resolved.

Example:

```
vopt +cover=bcest+/top/dut1. +cover=f+/top/dut1/fsm1 +cover=x+pla
```

This command enables branch, condition, expression, statement and toggle coverage for instance */top/dut1* and all its children, down to the leaf level. It also turns on FSM coverage for only the */top/dut1/fsm1* instance. Finally, it enables extended toggle coverage for all instances of design unit *pla*.

- -nocovershort

  (optional) Disables short circuiting of expressions when coverage is enabled. Short circuiting is enabled by default.

- -coverclkoptbuiltins | -nocoverclkbuiltins

  (optional) Enables/Disables clkOpt optimization builtins for code coverage. Used to enable or disable code coverage for VHDL simple builtin flops. You can customize the default behavior with the CoverClkOptBuiltins variable in the *modelsim.ini* file.

- -coverdeglitch {“<n> <time_unit>”}

  (optional) Enables deglitching of statement, branch, condition and expression coverage in combinational unclocked process/always blocks, concurrent (VHDL) and continuous (SV) assignments, where:

  [n] — A string specifying the number of time units.

  <time_unit> — (required if “n” is anything other than “0”) fs, ps, ns, us, ms, or sec.

  Quotes (“”) are required if a space exists between the two arguments (for example, 2ps or “2 ps”).

  If a process or a continuous assignment is evaluated more than once during any period of length <period>, only the last execution during that period is added into the coverage data for that process/continuous assignment. If a deglitch period of zero is specified, only the last delta cycle pass is counted toward the coverage data. For a new pass to be counted, it must occur at a time greater than the previous pass plus the deglitch period. You can customize the default behavior with the CoverDeglitchOn and CoverDeglitchPeriod variables in the *modelsim.ini* file.

- -coverenhanced

  (optional) Enables non-critical functionality which might change the appearance or content of coverage metrics. This argument only has an effect in letter releases (10.0a, 10.0b, and so on). In major releases (10.0, 10.1, and so on), all coverage enhancements present in previous letter release streams are enabled by default, and -coverenhanced is no longer necessary to enable these enhancements. Bug fixes important to the correctness of coverage numbers are always enabled by default, with no need for -coverenhanced. Since the exact nature of -coverenhanced varies from release to release, the details of the enhancements it enables are present in the product release notes rather than in the Command Reference. For these details, search the release notes using the string "coverenhanced".

- -coverexcludedefault

  (optional) Excludes the following:

  - VHDL code coverage data collection for the OTHERS branch in both Case statements and Selected Signal Assignment statements. All forms of code coverage collection for statements contained in the OTHERS branch will be excluded. Also excludes the allfalse branch (if any) associated with the case statement.

  - Verilog/SV code coverage data collection for the default branch in case statements. All forms of code coverage collection for statements contained in the default case statement will be excluded.

- -coverfec

  (optional) Enables focused expression coverage (FEC) for coverage collection. By default, only FEC coverage statistics are disabled for collection. You can customize the default behavior with the CoverFEC variable in the *modelsim.ini* file.

- -coveropt <opt_level>

  (optional) Overrides the default level of optimization for the current run only. <opt_level> designates the optimization level, as follows:

  1 — Turns off all optimizations that affect coverage reports.

  2 — Allows optimizations that provide large performance improvements by invoking sequential processes only when the data changes. This setting may result in major reductions in coverage counts.

  3 — (default) Allows all optimizations in 2, and allows optimizations that may remove some statements. Also allows constant propagation and VHDL subprogram inlining.

  4 — Allows all optimizations in 2 and 3, and allows optimizations that may remove major regions of code by changing assignments to built-ins or removing unused signals. It also changes Verilog gates to continuous assignments and optimizes Verilog expressions. Allows VHDL subprogram inlining. Allows VHDL flip-flop recognition.

  The default optimization level is 3. You can edit the CoverOpt variable in the *modelsim.ini* file to change the default.

- -coverrec

  (optional) Enables rapid expression coverage (REC) for coverage collection. By default, REC coverage statistics are enabled for collection. You can customize the default behavior with the CoverREC variable in the *modelsim.ini* file.

- -coverreportcancelled

  (optional) Enables code coverage reporting of branch conditions that have been optimized away due to a static or null condition. The line of code is labeled EA in the hits column of the Source Window and EBCS in the hits column of a Coverage Report. You can also set this with the CoverReportCancelled *modelsim.ini* variable.

- -coverudp

  (optional) Enables UDP coverage for coverage collection. By default, only FEC coverage statistics are enabled for collection. You can customize the default behavior with the CoverUDP variable in the *modelsim.ini* file.

- -debug[,viseffort=full | high | low]

  (optional) Enables increased visibility into VHDL, Verilog, and SystemVerilog designs.

  viseffort — (optional) Specifies the degree of visibility into a design after optimization with vopt. Requires one of the following options:

full — Enables 100% visibility. Disables dead-logic elimination. Designs with race conditions may simulate differently than fully optimized simulation runs.

high — (default) Enables 90% to 100% debug visibility depending on the design.

low — Fully optimizes the simulation. Object visibility is reduced to those objects visible after full optimization.

- -debugdb

  (optional) Enables advanced debugging features. Performs a pre-simulation debug analysis of the sequential and combinatorial elements in your design. The results are added to the optimized design unit and picked up when a simulation is run for later causality tracing and schematic functional representation.

- -debugCellOpt

  (optional) Produces Transcript window output that identifies why certain cells within the design were not optimized.

- -deferSubpgmCheck

  (optional) Forces the compiler to report array indexing and length errors as warnings (instead of as errors) when encountered within subprograms. Subprograms with indexing and length errors that are invoked during simulation cause the simulator to report errors, which can potentially slow down simulation because of additional checking.

- -deglitchalways | -nodeglitchalways

  Controls the behavior related to zero-delay oscillations among always_comb and always @* cominatorial logic blocks, as well as regular always blocks, that produce glitches on the variables they write.

  -deglitchalways — (default) Reduces the incidents of zero delay oscillations among the affected blocks.

  -nodeglitchalways — Disables the functionality. A side effect of this behavior is that time zero races involving the glitch-producing always blocks may resolve in a different order.

- +delay_mode_distributed

  (optional) Disables path delays in favor of distributed delays. Refer to “Delay Modes” for details.

- +delay_mode_path

  (optional) Sets distributed delays to zero in favor of using path delays.

- +delay_mode_unit

  (optional) Sets path delays to zero and non-zero distributed delays to one time unit.

- +delay_mode_zero

  (optional) Sets path delays and distributed delays to zero.

- -designfile <filename>

  (optional) Creates the design file required by Mentor Visualizer Debug Environment.

  <filename> — (required) A user specified string. May include a path specifying where to save the file.

- -dpiforceheader

  (optional) Forces the generation of a DPI header file even if it will be empty of function prototypes.

- -dpiheader <filename>

  (optional) Generates a header file that may then be included in C source code for DPI import functions. Refer to "DPI Use Flow" for additional information.

- -enablescstdout

  (optional) Enables the reporting of messages from the SystemC tasks cout, printf and fprintf to stdout during the execution of vopt. Also, reports a warning when a VPI routine is called in a SystemC constructor. This behavior is suppressed by default. However, information printed to stderr will always be displayed.

- -error <msg_number>[,<msg_number>,...]

  (optional) Changes the severity level of the specified message(s) to "error." Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -extendedtogglemode 1|2|3

  (optional) Changes the level of support for extended toggles. The levels of support are:

  1 — 0L->1H & 1H->0L & any one 'Z' transition (to/from 'Z')

  2 — 0L->1H & 1H->0L & one transition to 'Z' & one transition from 'Z'

  3 — 0L->1H & 1H->0L & all 'Z' transitions

  Edit the ExtendedToggleMode variable in the *modelsim.ini* file to set a permanent default.

- (-F | -file | -f) <filename>

  (optional) -f, -file and -F: each specifies an argument file with more command-line arguments, allowing complex argument strings to be reused without retyping. Nesting of -F, -f and -file commands is allowed. Allows gzipped input files.

  With -F only: relative file names and paths within the arguments file <filename> are prefixed with the path of the arguments file when lookup with relative path fails. Refer to the section "Argument Files" for more information.

- -fecudpeffort {1 | 2 | 3}

  (optional) The recommended option to set the level of effort for FEC or UDP. Levels are:

  1 — (default) (low) Only small expressions or conditions are considered for coverage.

  2 — (medium) Bigger expressions and conditions considered for coverage.

3 — (high) Very large expressions and conditions considered for coverage.

Increasing the effort level includes more expressions for coverage, but also increases the compile time. You can also enable this by setting the FecUdpEffort variable in the *modelsim.ini* file.

- -fatal <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "fatal." Edit the fatal variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -floatgenerics[+<selection>[+<recurse_level>|.]]

  (optional) Do not hold generic values constant during optimization, which enables successful use of the vsim -g/G options. Use of this argument causes finite state machines (FSMs) containing these floating parameters not to be recognized.

  +<selection>[+<recurse_level>|.] — enables access for specific Verilog design objects and/or regions.

  <selection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

  <library> — Any Verilog, SystemVerilog or VHDL library.

  <primary> — Specifies the design unit name.

  <secondary> — Specifies VHDL architecture.

  2. <instance_path> — A path to an instance.

  3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

Note

Glob wildcards ("*" and "?") are accepted only for the specification of design unit(s).

<recurse_level> — Specifies the number of recursion levels that selection will occur downward from the current level of the specified module or instance. This can be any integer between 0 and 128, where a value of 0 applies full recursion.

. (period) — Indicates that selection occurs recursively downward from the specified module or instance.

- All modules — do not include the *selection* argument.
- Single module — include only a module name as the *selection* argument.

  ```
  -floatgenerics+top
  ```

- Single module and all instances below — append a period to the module name.

  ```
  -floatgenerics+top.
  ```

- Specific instance — include a leading slash and the instance path

```
-floatgenerics+/top/mid1/bottom/
```

- Specific instance and all instances below — include a leading slash and the instance path and append a period to the path.

```
-floatgenerics+/top/mid1/bottom/.
```

- Specific instance within a submodule — do not include a leading slash with the instance path.

```
-floatgenerics+mid/bottom/
```

Refer to the section "Optimization of Parameters and Generics" for more information.

This command is fully equivalent to -floatparameters, therefore you can use them interchangeably.

- -floatparameters[[+<selection>[+<recurse_level>|.]] | [+<leafparameter>]]

  (optional) Instructs the tool to not lock down parameter values during optimization, which enables successful use of the vsim -g/G options. Use of this argument can cause FSMs containing these floating parameters not to be recognized.

  +<selection>[+<recurse_level>|.] — enables access for specific Verilog design objects and/or regions.

  <selection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

  <library> — Any Verilog, SystemVerilog or VHDL library.

  <primary> — Specifies the design unit name.

  <secondary> — Specifies VHDL architecture.

  2. <instance_path> — A path to an instance.

  3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

Note

Glob wildcards (“*” and “?”) are accepted only for the specification of design unit(s).

<recurse_level> — Specifies the number of recursion levels that selection will occur downward from the specified module or instance.

. (period) — Indicates that selection occurs recursively downward from the specified module or instance.

+<leafparameter> — floats the specified parameter across the design.

- All modules — do not include the *selection* argument.
- Single module — include only a module name as the *selection* argument.

```
-floatparameters+top
```

- Single module and all instances below — append a period to the module name.

```
-floatparameters+top.
```

- Specific instance — include a leading slash and the instance path

```
-floatparameters+/top/mid1/bottom/
```

- Specific instance and all instances below — include a leading slash and the instance path and append a period to the path.

```
-floatparameters+/top/mid1/bottom/.
```

- Specific instance within a submodule — do not include a leading slash with the instance path.

```
-floatparameters+mid/bottom/
```

Refer to the section "Optimization of Parameters and Generics" for more information.

This command is fully equivalent to -floatgenerics, therefore you can use them interchangeably.

- -[no]fsmdebug [=[-]<accesscodes>][+<objectselection>]

  (optional) Enable or disable access to finite state machines for debugging.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

  <objectselection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

  <library> — Any Verilog, SystemVerilog, or VHDL library.

  <primary> — Specifies the design unit name.

  <secondary> — Specifies VHDL architecture.

  2. <instance_path> — A path to an instance.

  3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

- -fsmimplicittrans

  (optional) Enables recognition of implied same state transitions.

- -fsmmultitrans

  (optional) Enables detection and reporting of multi-state transitions when used with the +cover=f argument for vcom/vlog or vopt. Another term for this is FSM sequence coverage.

- -fsmresettrans | -nofsmresettrans

  (optional) Toggles recognition of synchronous or asynchronous reset transitions, and is on by default. This includes/excludes reset transitions in coverage results.

- -fsmsingle | -nofsmsingle

  (optional) Enables the recognition of VHDL FSMs where the current state variable of type std_logic, bit, boolean, or single-bit std_logic_vector/bit_vector and Verilog single-bit FSMs.

- -fsmverbose [b | t | w]

  (optional) Provides information about FSMs detected, including state reachability analysis.

  b — displays only basic information.

  t — displays a transition table in addition to the basic information.

  w — displays any warning messages in addition to the basic information.

  When you do not specify an argument, this switch reports all information to vopt message 1947.

```
# ** Note: (vopt-1947)   FSM RECOGNITION INFO
#      Fsm detected in : ../fpu/rtl/vhdl/serial_mul.vhd
#      Current State Variable : s_state :
../fpu/rtl/vhdl/serial_mul.vhd(76)
#      Clock : clk_i
#      Reset States are: { waiting , busy }
#      State Set is : { busy , waiting }
#      Transition table is
#         -----------------------------------------
#         busy      =>   waiting Line : (114 => 114)
#         busy      =>   busy    Line : (111 => 111)
#         waiting   =>   waiting Line : (120 => 120) (114 => 114)
#         waiting   =>   busy    Line : (111 => 111)
#         -----------------------------------------
```

  When you do not specify this switch, vopt reports only that an FSM was detected in vopt message 143.

```
# ** Note: (vopt-143) Detected '1' FSM/s in design unit
'serial_mul.rtl'.
```

- -g <Name>=<Value> …

  (optional) Assigns a value to all specified VHDL generics and Verilog parameters that have not received explicit values in generic maps, instantiations, or via defparams (such as top-level generics/parameters and generics/parameters that would otherwise receive their default values).

  For more information on this switch, refer to the longer description under vsim -g <Name>=<Value> …

Refer to the section "Optimization of Parameters and Generics" in the User's Manual for additional information.

Limitation: In general, generics/parameters of composite type (arrays and records) cannot be set from the command line. However, you can set string arrays, std_logic vectors, and bit vectors if they can be set using a quoted string. For example,

```
-g strgen="This is a string"
-g slv="01001110"
```

The quotation marks must make it into vsim as part of the string because the type of the value must be determinable outside of any context. Therefore, when entering this command from a shell, put single quotes (') around the string. For example:

```
-g strgen='"This is a string"'
```

If working within the Questa SIM GUI, you would enter the command as follows:

```
{-g strgen="This is a string"}
```

You can also enclose the value escaped quotes (\"), for example:

```
-g strgen=\"This is a string\"
```

- -G <Name>=<Value> …

  (optional) Same as -g (refer above) except that it will also override generics/parameters that received explicit values in generic maps, instantiations, or from defparams. This argument is the only way for you to alter the generic/parameter, such as its length, (other than its value) after the design has been loaded.

  If <value> is a string, you must enclose it in escaped quotes (\"), for example:

```
-G filename=\"a.in\"
```

  Refer to the section "Optimization of Parameters and Generics" for more information.

- -gblso <shared_obj>[,<shared_obj> …]

  (optional) Open the specified shared object(s) with global symbol visibility. Essentially all data and functions are exported from the specified shared object and are available to be referenced and used by other shared objects. If you specify multiple, comma-separated, shared objects, they will be merged internally and then loaded as a single shared object. You can also specify this argument with the GlobalSharedObjectList variable in the *modelsim.ini* file.

- -hazards

  (optional) Detects event order hazards involving simultaneous reading and writing of the same register in concurrently executing processes. You must also specify this argument when you simulate the design with vsim. Refer to "Hazard Detection" for more details.

Note

Enabling -hazards implicitly enables the -compat argument. As a result, using this argument may affect your simulation results.

- -incr | -noincr

  (optional) Defines whether vopt reuses design elements or not.

  -incr — Instructs vopt to only optimize design elements that have changed since a previous optimization. Default

  -noincr — Instructs vopt to optimize all design elements, even if they have not changed since a previous optimization.

- +initmem[=<spec>][+{0 | 1 | X | Z}][+<selection>[+<recurse_level>|.]] | +noinitmem[+<selection>[+<recurse_level>|.]]

  (optional) Allows you to enable or disable the initialization of memories.

Note

+initmem accepts any of the following arguments, whereas +noinitmem accepts only the +<selection> arguments.

=<spec> — (optional) identifies the types to be initialized: valid with +initmem only.

If you do not specify this option, vlog initializes fixed-size arrays of all these types, where fixed-size arrays may have any number of packed or unpacked dimensions.

<spec> can be one or more of the following:

r — register/logic, integer, or time types (four-state integral types).

b — bit, int, shortint, longint, or byte types (two-state integral types).

e — enum types.

You must also add the enum's base type to the initialization specification. If you choose static initialization for an enum type variable with value 0, 1, X, or Z, the simulator assigns that value to the variable, whether it is a valid value or not. If you choose random initialization for an enum type variable, the simulator generates a random number and uses the (random_number % num_valid_enum_values)th entry of the enum literals to initialize it.

+{0 | 1 | X | Z} — (optional) specifies the initialization value to use for all the elements of a memory: valid with +initmem only. For example, the +initmem+1 option initializes "reg [7:0] m" to value 'hff. For two-state datatypes, X and Z will map to 0.

If you do not specify the value, initmem will randomly initialize this memory to a 2-state value. The seed for randomization can then be controlled by vsim command line option +initmem+<seed>, where <seed> is a 32-bit number.

These options do not result in scheduling events at time 0. The specified memories are initialized to these values. So, if you set these memories to the same values that +initmem initialized them to, events will not be generated. The exception is that of sequential UDPs that are initialized in the NBA region of time 0.

+<selection>[+<recurse_level>|.] — enables access for specific Verilog design objects and/or regions.

<selection> — Can take one of the following forms:

1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

<library> — Any Verilog, SystemVerilog or VHDL library.

<primary> — Specifies the design unit name.

<secondary> — Specifies VHDL architecture.

2. <instance_path> — A path to an instance.

3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

Note

Glob wildcards (“*” and “?”) are accepted only for the specification of design unit(s).

<recurse_level> — Specifies the number of recursion levels that selection will occur downward from the current level of the specified module or instance. This can be any integer between 0 and 128, where a value of 0 applies full recursion.

. (period) — Indicates that selection occurs recursively downward from the specified module or instance.

If you do not specify this argument, the initialization is provided to the entire design.

If initialization is recursively extended to descendants of a given design unit, and a descendant has an explicit initialization specification applied to it, the child's initialization specification overrides the parent's initialization specification. The initialization specifications of the parent and child are not merged together (as is done with +acc options). The default initialization state for top modules is "no_init", provided a top module does not have an explicit initialization specification applied to it.

- All modules — do not include the *selection* argument.
- Single module — include only a module name as the *selection* argument.

  ```
  +initmem=rb+1+top
  ```

- Single module and all instances below — append a period to the module name.

  ```
  +initmem=rb+1+top.
  ```

- Specific instance — include a leading slash and the instance path

  ```
  +initmem=rb+1+/top/mid1/bottom/myReg
  ```

- Specific instance and all instances below — include a leading slash and the instance path and append a period to the path.

  ```
  +initmem=rb+1+/top/mid1/bottom/myReg.
  ```

This argument initializes static variables in any scope (package, $unit, module, interface, generate, program, task, function). However, it does not affect:

- automatic variables

- dynamic variables
- members of dynamic variables
- artificially generated variables, such as #randstate#

In case +initmem options are specified to vlog and vopt command line , the priority of the specifications are as follows:

```
1) design unit or instance specific options specified during vopt –
   for example, vopt ... +initmem+1+top.foo
2) options specified during vlog — for example, vlog ... +initmem+0
3) global options specified during vopt — vopt … +initmem+Z
```

This argument will not override any variable declaration assignment, such as:

```
reg r = 1'b0
```

- -initoutcompositeparam

  (optional) Causes initialization of VHDL subprogram parameters for array and record types when the subprogram is executed in designs compiled with LRM 1076-2002 and earlier. This argument forces the output parameters to their default initial ("left") values when entering a subprogram. By default, -initoutcompositeparam is enabled for designs compiled with vcom -2008 and later. You can also enable this by setting the InitiOutCompositeParam variable to 1 in the *modelsim.ini* file.

- -noinitoutcompositeparam

  (optional) Disables initialization of VHDL subprogram parameters for array and record types when the subprogram is executed in designs compiled with LRM 1076-2002 and earlier. By default, designes compiled with LRM 1076-2008 and later do not initialize subprogram parameters for array and record types when the subprogram is executed. You can also disable initialization of subprogram parameters for array and record types by setting the InitiOutCompositeParam variable to 2 in the *modelsim.ini* file.

- +initreg[=<spec>][+{0 | 1 | X | Z}][+<selection>[+<recurse_level>|.]] | +noinitreg[+<selection>[+<recurse_level>|.]]

  (optional) Allows you to enable or disable the initialization of registers.

Note

+initreg accepts any of the following arguments, whereas +noinitreg accepts only the +<selection> arguments.

=<spec> — (optional) identifies the types to be initialized: valid with +initreg only.

If you do not specify this option, vlog initializes variables of all these types.

<spec> can be one or more of the following:

r — register/logic, integer, or time types (four-state integral types).

Notifier registers are not initialized by the +initreg option.

b — bit, int, shortint, longint, or byte types (two-state integral types).

e — enum types.

You must also add the enum's base type to the initialization specification. If you choose static initialization for an enum type variable with value 0, 1, X, or Z, the simulator assigns that value to the variable, whether it is a valid value or not. If you choose random initialization for an enum type variable, the simulator generates a random number and uses the (random_number % num_valid_enum_values)th entry of the enum literals to initialize it.

u — sequential UDPs.

If a sequential UDP contains an "initial" statement, that initial value overrides all +initreg-related functionality. For other sequential UDPs, the +initreg option takes effect as described for regular variables. In case a sequential UDP does not contain an "initial" statement, and it wasn't compiled with +initreg in effect, the UDPs initial value will be taken from its instantiating parent scope (provided that scope has +initreg options in effect).

+{0 | 1 | X | Z} — (optional) specifies the initialization value to use for all the bits of a register: valid with +initreg only. For example, the +initreg+1 option initializes "reg [7:0] r" to value 'hff. For two-state datatypes, X and Z will map to 0.

If you do not specify the value, initreg will randomly initialize this register to a 2-state value. The seed for randomization can then be controlled by vsim command line option +initreg+<seed>, where <seed> is a 32-bit number.

These options do not result in scheduling events at time 0. The specified registers are initialized to these values. So, if you set these registers to the same values that +initreg initialized them to, events will not be generated. The exception is that of sequential UDPs that are initialized in the NBA region of time 0.

+<selection>[+<recurse_level>|.] — enables access for specific Verilog design objects and/or regions.

<selection> — Can take one of the following forms:

1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

<library> — Any Verilog, SystemVerilog or VHDL library.

<primary> — Specifies the design unit name.

<secondary> — Specifies VHDL architecture.

2. <instance_path> — A path to an instance.

3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

Note

Glob wildcards (“*” and “?”) are accepted only for the specification of design unit(s).

<recurse_level> — Specifies the number of recursion levels that selection will occur downward from the current level of the specified module or instance. This can be any integer between 0 and 128, where a value of 0 applies full recursion.

. (period) — Indicates that selection occurs recursively downward from the specified module or instance.

If you do not specify this argument, the initialization is provided to the entire design.

If initialization is recursively extended to descendants of a given design unit, and a descendant has an explicit initialization specification applied to it, the child's initialization specification overrides the parent's initialization specification. The initialization specifications of the parent and child are not merged together (as is done with +acc options). The default initialization state for top modules is "no_init", provided a top module does not have an explicit initialization specification applied to it.

- All modules — do not include the *selection* argument.
- Single module — include only a module name as the *selection* argument.

  ```
  +initreg=rb+1+top
  ```

- Single module and all instances below — append a period to the module name.

  ```
  +initreg=rb+1+top.
  ```

- Specific instance — include a leading slash and the instance path

  ```
  +initreg=rb+1+/top/mid1/bottom/myReg
  ```

- Specific instance and all instances below — include a leading slash and the instance path and append a period to the path.

  ```
  +initreg=rb+1+/top/mid1/bottom/myReg.
  ```

This argument initializes static variables in any scope (package, $unit, module, interface, generate, program, task, function). However, it does not affect:

- automatic variables
- dynamic variables
- members of dynamic variables
- artificially generated variables, such as #randstate#

In case +initreg options are specified to vlog and vopt command line , the priority of the specifications are as follows:

```
1) design unit or instance specific options specified during vopt –
   for example, vopt ... +initreg+1+top.foo
2) options specified during vlog – for example, vlog ... +initreg+0
3) global options specified during vopt – vopt … +initreg+Z
```

This argument will not override any variable declaration assignment, such as:

```
reg r = 1'b0
```

- -j {0 | 1| <n>}

  (optional) Setting this argument to 0 ensures that vopt preserves the proper order of vopt output messages that might otherwise appear out of order due to forked-off sub-processes being performed in parallel. Setting the argument to 1 limits vopt to one process (this is the same as as using 0). Setting the argument to a positive integer greater than one (n > 1) can improve vopt performance. The default behavior is vopt uses a heuristic to set the number of parallel processes.

- -keep_delta

  (optional) Disables optimizations that remove delta delays.

  Delta delays result from zero delay events. Those events are normally processed in the next iteration or "delta" of the current timestep. Vopt implements optimizations that can remove delta delays and process an event earlier.

- -libertyfiles=<filename> [,<filename>,<filename>,...]

  (optional) Specifies one or more Liberty cell library files that contain liberty logic cell definitions. Enables schematic viewing and causality analysis of liberty cells. Refer to MTI_LIBERTY_PATH environment variable and Liberty Library Models for more information

  <file_name> — A comma-separated list of liberty file names.

- -L <libname>

  (optional) Searches the specified resource library for precompiled modules. The library search options you specify here must also be specified when you run the vsim command.

- -Lf <libname>

  (optional) Same as -L, but the specified library is searched before any ‘uselib directives. (Refer to “Library Usage” and “Verilog-XL Compatible Compiler Arguments” for more information.)

- -[no]linedebug [=[-]<accesscodes>][+<objectselection>]

  (optional) Enable or disable line debugging.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

  <objectselection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a

design unit name.

<library> — Any Verilog, SystemVerilog, or VHDL library.

<primary> — Specifies the design unit name.

<secondary> — Specifies VHDL architecture.

2. <instance_path> — A path to an instance.

3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

- -lowercasepslpragma

  (optional) Forces the optimization utility to accept only lower case (embedded) PSL pragmas.

- -lrmclassinit

  (optional) Changes initialization behavior to match the SystemVerilog specification (per IEEE Std 1800-2007) where all superclass properties will be initialized before any subclass properties.

- -logfile <filename> | -l <filename>

  (optional) Generates a log file of the compile.

  -logfile <filename> — Saves transcript data to <filename>. Can be abbreviated to -l <filename>. Overrides the default transcript file creation set with the the TranscriptFile or BatchTranscriptFile *modelsim.ini* variables. You can also specify “stdout” or “stderr” for <filename>.

- +maxdelays

  (optional) Selects maximum delays from the "min:typ:max" expressions. If preferred, you can defer delay selection until simulation time by using vsim -sdfmax.

  If you specify the +mindelays, +typdelays, or +maxdelays flag with vopt, and specify a different flag with vsim, the simulation will be able to use the delay value based upon the flag specified with vopt. You must specify vsim -novopt to force the simulator to use the delay flag specified with vsim.

- +mindelays

  (optional) Selects minimum delays from the "min:typ:max" expressions. If preferred, you can defer delay selection until simulation time by using vsim -sdfmin.

  If you specify the +mindelays, +typdelays, or +maxdelays flag with vopt, and specify a different flag with vsim, the simulation will be able to use the delay value based upon the flag specified with vopt. You must specify vsim -novopt to force the simulator to use the delay flag specified with vsim.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable. Specifies either

an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -mti_trace_vlog_calls

  (optional) Enables viewing of SystemVerilog class contents in the Wave window.

- -noconstimmedassert

  (optional) Turns off the display of immediate assertions with constant expressions in the GUI, in reports, and in the UCDB. By default, immediate assertions with constant expressions are displayed. You may also set the ShowConstantImmediateAsserts variable in the vopt section of the *modelsim.ini* file to 0 (off).

- -nocoverexcludedefault

  (optional) Allows VHDL branch code coverage data collection for the OTHERS branch in both Case statements and Selected Signal Assignment statements, and Verilog/SV branch code coverage data collection for the default branch in case statements.

- -nocoverfec

  (optional) Disables focused expression coverage (FEC) for coverage collection. By default, FEC coverage statistics are disabled for collection. You can customize the default behavior with the CoverFEC variable in the *modelsim.ini* file.

- -nocoverrec

  (optional) Disables rapid expression coverage (REC) for coverage collection. By default, REC coverage statistics are enabled for collection. Disabling REC collection converts non-masking conditions in FEC tables to matching input patterns. You can customize the default behavior with the CoverREC variable in the *modelsim.ini* file.

- -nocoverudp

  (optional) Prevents UDP coverage from being enabled for coverage collection. By default, only FEC coverage statistics are enabled for collection. You can customize the default behavior with the CoverUDP variable in the *modelsim.ini* file.

- -nocreatelib

  (optional) Stops automatic creation of missing work libraries and reverts back to 10.3x and earlier version behavior. Overrides the CreateLib modelsim.ini variable.

- -nodebug[=ports | =pli | =ports+pli]

  (optional) Hides, within the GUI and other parts of the tool, the internal data of all compiled design units.

  -nodebug — The switch, specified in this form, does not hide ports, due to the fact that the port information may be required for instantiation in a parent scope.

  The design units' source code, internal structure, registers, nets, and so on, will not display in the GUI. In addition, none of the hidden objects may be accessed through the Dataflow or Schematic windows, or with commands. This also means that you

cannot set breakpoints or single step within this code. It is advised that you not compile with this switch until you are done debugging.

Note that this is not a speed switch like the “nodebug” option on many other products. Use the vopt command to increase simulation speed.

-nodebug=ports — additionally hides the ports for the lower levels of your design; it should be used only to compile the lower levels of the design. If you hide the ports of the top level you will not be able to simulate the design.

Do not use the switch in this form when the parent is part of a vopt Preoptimized Design Unit (black-box) flow or for mixed language designs, especially for Verilog modules to be instantiated inside VHDL.

-nodebug=pli — additionally prevents the use of pli functions to interrogate individual modules for information.

You should be aware that this form will leave a "nodebug" module untraversable by PLI.

-nodebug=ports+pli — you can combine the behavior of =ports and =pli in this manner.

This functionality encrypts entire files. The `protect compiler directive allows you to encrypt regions within a file.

- -noexcludeternary

  (optional) Disables the automatic exclusion of UDB coverage data rows resulting from ternary expressions for the optimized design. Normal operation for code coverage is to include rows corresponding to the case where two data inputs are the same, and the select input is a “don’t care”. To disable this automatic exclusion for the entire design, use “vsim -noexcludeternary” instead.

- -nofprangecheck

  (optional) Disables range checks on floating type values only.

- -nofsmxassign

  (optional) Disables recognition of FSMs containing x assignment.

- +nolibcell

  +libcell — (optional) Treats all modules found and compiled by source library search as though they contained a ‘celldefine compiler directive, thus marking them as cells (refer to the -v and -y arguments of vlog, which enable source library search). Using the +libcell argument matches historical behavior of Verilog-XL with respect to source library search.

  +nolibcell — (default) Disables treating all modules found and compiled by source library search as though they contained a ‘celldefine compiler directive. That is, this argument restores the default library search behavior if you have changed it using the +libcell | +nolibcell argument.

- -nologo

  (optional) Disables the startup banner.

- -noscelab

  (optional) Disables vopt's elaboration of SystemC design hierarchy and HDL sub-hierarchy, if any exists. This switch can be used when you do not want to elaborate SystemC design in vopt, such as the case where a SystemC constructor depends on a runtime state variable. Recommended for use when the SystemC design does not instantiate an HDL design.

- -noscmainscopename

  (optional) When design units are instantiated under sc_main, Questa SIM creates a scope called sc_main which is then appended to the hierarchical path returned by the name() function, rendering the design incompatible with OSCI. The -noscmainscopename option strips out the sc_main scope name from the output of the name() function.

- -nosparse

  (optional) Turn off sparse memory processing for memories that would otherwise be implemented with sparse storage. Identifies which memories are considered "not sparse," which causes Questa SIM to override the rules for allocating storage for memory elements only when necessary. If you use -nosparse on a given memory, Questa SIM will simulate the memory normally. Refer to Sparse Memory Modeling for more information.

- +nosparse[+<selection>[+<recurse_level>|.]]

  (optional) Identifies which memories are considered "not sparse", which instructs the tool to override the rules for allocating storage for memory elements only when necessary.

  If you use +nosparse on a given memory, the tool will simulate the memory normally. Refer to Sparse Memory Modeling for more information.

  +<selection>[+<recurse_level>|.] — enables access for specific Verilog design objects and/or regions.

  <selection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

  <library> — Any Verilog, SystemVerilog or VHDL library.

  <primary> — Specifies the design unit name.

  <secondary> — Specifies VHDL architecture.

  2. <instance_path> — A path to an instance.

  3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

Note

Glob wildcards ("*" and "?") are accepted only for the specification of design unit(s).

<recurse_level> — Specifies the number of recursion levels that selection will occur downward from the current level of the specified module or instance. This can be any integer between 0 and 128, where a value of 0 applies full recursion.

. (period) — Indicates that selection occurs recursively downward from the specified

module or instance.

- All modules — do not include the *selection* argument.
- Single module — include only a module name as the *selection* argument.

  ```
  +nosparse+top
  ```

- Single module and all instances below — append a period to the module name.

  ```
  +nosparse+top.
  ```

- Specific instance — include a leading slash and the instance path

  ```
  +nosparse+/top/mid1/bottom/myReg
  ```

- Specific instance and all instances below — include a leading slash and the instance path and append a period to the path.

  ```
  +nosparse+/top/mid1/bottom/myReg.
  ```

- Specific instance within a submodule — do not include a leading slash with the instance path.

  ```
  +nosparse+mid/bottom/myReg
  ```

- +nospecify

  (optional) Disables specify path delays and timing checks.

- -note <msg_number>[,<msg_number>,...]

  (optional) Changes the severity level of the specified message(s) to "note." Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information.

- +notimingchecks

  (optional) Removes all timing check entries from the design as it is parsed. To disable checks on individual instances, use the tcheck_set command.

  Specifying vopt +notimingchecks or -GTimingChecks=<FALSE/TRUE> will fix the TimingChecksOn generic value in VITAL models to FALSE for simulation. As a consequence, using vsim +notimingchecks at simulation may not have any effect on the simulation depending on the optimization of the model.

- +nowarn<CODE>

  (optional) Disables warning messages in the category specified by <CODE>. Warnings that can be disabled include the <CODE> name in square brackets in the warning message. For example,

  ```
  ** Warning: test.v(15): [RDGN] - Redundant digits in numeric
  literal.
  ```

  This warning message can be disabled by specifying +nowarnRDGN.

- -nowarn <category_number>

  (optional) Selectively disables a category of warning message. Multiple -nowarn switches are allowed. Warnings may be disabled for all compiles via the Main window **Compile > Compile Options** menu command or the *modelsim.ini* file (refer to modelsim.ini Variables).

  The warning message categories are described in Table 2-16:

Table 2-16. Warning Message Categories for vopt -nowarn

Category number	Description
1	unbound component
2	process without a wait statement
3	null range
4	no space in time literal
5	multiple drivers on unresolved signal
6	VITAL compliance checks
7	VITAL optimization messages
8	lint checks
9	signal value dependency at elaboration
10	VHDL-1993 constructs in VHDL-1987 code
11	PSL warnings
12	non-LRM compliance in order to match Cadence behavior
13	constructs that coverage can't handle
14	locally static error deferred until simulation run
15	SystemVerilog assertions using local variable

- +num_opt_cell_conds+<value>

  (optional) Restricts gate-level optimization capacity for accepting cells with I/O path and timing check conditions.

  value — integer between 32 and 1023, inclusive. where the default value is 1023.

- -O0 | -O1 | -O4 | -O5

  (optional) Lower the optimization to a minimum with -O0 (capital oh zero). Use this to work around bugs, increase your debugging visibility on a specific cell, or when you want to place breakpoints on source lines that have been optimized out.

  Please refer to "Optimizing Designs with vopt" in the User's Manual for detailed information on using vopt to perform optimization.

- (optional) Enable some optimization with -O1.
- (default) Enable most optimizations with -O4.
- Enable maximum optimization with -O5. Questa SIM attempts to optimize loops and prevents variable assignments in situations where a variable is assigned but is not actually used.

- -ocf <filename>

  (optional) Specifies an OCF (*.ocf*) or OCM (*.ocm*) file that controls the visibility for the simulation, based on using the learn flow. Refer to "Preserving Design Visibility with the Learn Flow" in the User's Manual for more information. Access flags from OCF files can be applied to objects under Verilog unnamed generate scopes as well as objects specified in SignalSpy system tasks.

- -optionset <optionset_name>

  (optional) Calls an optionset as defined in the *modelsim.ini file*. Refer to "Optionsets" for more information.

- -pa_behavlogfile <filename>

  (optional) Generates a list of all the non-synthesizable constructs found in design to a text file, specified by <filename>. This list identifies all the constructs that must be in always-on power domain.

- -pa_checks=<check_value>

Note

To specify more than one check_value, use the + operator as a separator. For example: vopt -pa_checks=rop+cp+a

Note

You can enable all power aware checks (static and dynamic) by specifying the -pa_checks=all argument. You can enable all dynamic power aware checks by specifying the -pa_checks=d argument or -pa_checks without any argument. You can enable all static power aware checks by specifying the -pa_checks=s argument.

(optional) Specifies the static or dynamic checks to perform.

<check_value> — one or more of the following arguments, separated by a plus sign (+). If you do no specify an argument, all dynamic checks will be tested.

rop — (Dynamic) States that there were retention signals mapped to the power domain. However, these signals were not asserted when the power was switched off.

rpo — (Dynamic) States that there was an error in the sequence of triggering of the retention and power signal. For retention to succeed, the power should be high. However, this check is triggered when that is not the case

rcs — (Dynamic) Certain Power Aware models require that the clock/latch enable must be at a certain value when retention takes place. This check displays error when this condition is not satisfied. If latch is enabled and can change its value, triggering retention can potentially cause race in the stored value. This is also a check against such conditions.

rsa — (Dynamic) Some Power Aware models require that the clock not toggle when the power is down. This check helps in monitoring this condition.

rtcon — (Dynamic) Flags a violation if the retention condition is not true at the time of retention enable.

rtcoff — (Dynamic) Flags a violation if the retention condition is not true at the time of retention disable.

rtctog — (Dynamic) Flags a violation if retention condition toggled during the retention period.

rtc — (Dynamic) Enables all retention condition checks: rtcon, rtcoff, and rtcog.

r — (Dynamic) Enable retention checking performed by all preceding retention arguments (those beginning with “r”).

icp — (Dynamic) Ensures isolation cell is clamping to correct clamp value specified in UPF file. Catches any functional issue in isolation cell inserted in RTL or applied isolation.

iep — (Dynamic) Detects conditions where the isolation signal is not enabled when the power to a power domain goes down.

idp — (Dynamic) Triggered when the isolation signal is enabled and the power to a power domain has already gone down.

iepcoa — (Dynamic) Detects violations when isolation control is not enabled when a driving power domain goes into any of the CORRUPT_* states and the receiving logic does not have a CORRUPT simstate.

idpcoa — (Dynamic) Detects violations when isolation control is disabled during the CORRUPT_* states when driving domain and receiving logic does not have a CORRUPT simstate.

ifc — (Dynamic) Ensures that when isolation is not applied, the value at isolation cell’s output is same as that at its input.

irc — (Dynamic) It is possible for the isolated ports to change value when the isolation is triggered. This check catches such error conditions at the negative and positive edges of isolation enable. (Not supported for UPF)

it — (Dynamic) Ensures that the isolated ports do not toggle when isolation is high. (Not supported for UPF)

ira — (Dynamic) Ensures that if an activation of isolation control occurs that it is required by power management behavior.

isa — (Dynamic) Some isolation strategies require that another reset signal be used to freeze the value of isolation. This check ensures that the isolated value is the same at posedge of reset and at posedge of isolation enable signal. (Not supported for UPF)

upc — (Dynamic) Checks for switching off of retention/isolation supplies during active retention/isolatio period (UPF only).

umi — (Dynamic) Catches failure occurring whenever source domain is OFF and sink domain is ON for a power domain crossing and if isolation strategy is not specified for that domain crossing.

i — (Dynamic) Enables isolation checking performed by all preceding isolation arguments. (For UPF, this value enables only iep and idp.)

uml — (Dynamic) Checks for missing level shifter; During simulation there will be an assertion failure whenever there is a difference between the operating voltage of source and sink domains.

uil — (Dynamic) Checks for incorrect level shifter; During simulation there will be an assertion failure whenever there is a mismatch in the shift direction of the level shifter specified at the domain boundary.

ul — (Dynamic) Enables all dynamic checks for level shifting.

t — (Dynamic) Enables the simulator to catch a condition where inputs to the power domain toggle even when the power domain is turned off.

cp — (Dynamic) Helps catch conditions when the power signal to any power domain gets corrupted during simulation.

p — (Dynamic) Identifies when each power domain is switched on or off.

pis — (Dynamic) Checks for illegal or undefined states on a PST or supply port.

npu — (Dynamic) Detects whether asynchronous (set/reset) controls for non-retention flip-flops are asserted at power-up.

ugc — (Dynamic) Checks for glitches (multiple value change on same time step) on controls of any power objects (isolation, retention, switches—UPF only).

smi — (Static) Checks for missing isolation cells for isolation strategies not specified on domain crossings where the source domain is relatively OFF with respect to the sink domain.

sri — (Static) Checks for redundant isolation cells for isolation strategies specified on domain crossings but isolation is not required. Also checks for redundant isolation cells placed when isolation is not required from driving power domain to sink power domain.

sii — (Static) Checks for incorrect isolation cells when isolation is required for power domain crossing, but isolation strategy is specified with set_isolation -no_isolation UPF command.

svi — (Static) Checks for required (valid) isolation cells and isolation strategies specified on domain crossings.

sni — (Static) Checks for unanalyzed isolation strategies on domain crossings where the power state table (PST) information is not sufficient to determine whether isolation is required.

sdi — (Static) Flags isolation cells as Not Inserted when isolation is not required for power domain crossings and isolation strategies specified with set_isolation -no_isolation UPF command.

si — (Static) Performs all static isolation checks (smi, sri, sii, svi, sni, sdi).

sml — (Static) Checks for any level shifter that is missing at the particular source to sink power domain.

sil — (Static) Checks for any level shifter that shifts in a particular direction (such as high-to-low) but the actual potential difference inferred from state table is in a different direction (such as low-to-high).

srl — (Static) Checks for any level shifter whose input and output supplies have no voltage difference. Questa SIM flags them as a redundant level shifter.

snl — (Static) Checks for a level shifter present at the boundary where there is no voltage information available. This could be possible due to a missing entry in the Power State Table. Questa SIM ignores such level shifters for multi-voltage analysis and flags them as Not Analyzed.

svl — (Static) Checks for valid level shifters that have matching voltage information and proper direction of shift.

sdl — (Static) Checks for a level shifter that has not been inserted because you have specified set_level_shifter -no_shift in the UPF file, where running a static analysis of power domain boundaries results in one of the following:

Level shifter not required — the report contains a counter labeled “Not inserted” that shows the number of level shifters found but not inserted.

Level shifter required — Questa SIM has determined a need for a level shifter and flags it as an incorrect level shifter.

sl — (Static) Enables all static checks for level shifting.

s — (Static) Enables all static checks for isolation and level shifting.

all — Enables all static and dynamic checks.

d — Enables all dynamic checks.

- -pa_checksoption=<value>

  (optional) Global settings that affect all static and dynamic checks set with the -pa_checks argument. Where <value> can be:

  assertionhierpath — any enabled dynamic checks will display the full hierarchical path of any failing checks.

- -pa_connectpgpin i | a | e | c

  (optional) Bypasses the connection of supply net using connect_supply_net UPF command to port when the port is missing in the verification model but it is a pg pin in the liberty model.

i — Ignores the connect_supply_net UPF command to these ports.

a — Ignores the connect_supply_net UPF command to these ports and disables the Power Aware simulation semantics of the parent instance of the port.

e — Flag error message for these ports (default).

c — Enable the creation of a port when connect_supply_net identifies a port which is missing in HDL but is specified as a pg_pin in a Liberty model.

- -pa_corrupt={real | integer} | -pa_nocorrupt={real | integer}

  (optional) Controls the corruption of various types in the design.

  -pa_corrupt=real — enables corruption of real types.

  -pa_nocorrupt=real — (default) disables corruption of real types.

  -pa_corrupt=integer — (default) enables corruption of integer types.

  -pa_nocorrupt=integer — disables corruption of integer types.

- -pa_defertop

  (optional) Change root of DUT hierarchy to a new pathname when you run simulation using the vsim -pa_top=<pathname> command. The result is that Questa SIM creates Power Aware dumps that you can change during simulation runtime without having to rerun vopt.

- -pa_dbgfindobj=<filename>

  Writes the results of all find_objects UPF commands to the specified <filename>.

- -pa_disable=<value>[+<value>]…

  (optional) Disables certain actions that Questa SIM performs during Power Aware simulation. You can use these values to control compatibility with previous releases.

  <value> — any of the following, in a plus (+) separated list.

  Refer to the section "Usage of vopt -pa_enable and -pa_disable" in the *Power Aware Simulation User's Manual* for a complete list of values.

Note

To specify more than one value, use the + operator between values.
For example: vopt -pa_disable=insertls+insertiso

- -pa_dumplibertydb=<database_pathname>

  (optional) Specifies a location for a user-defined database for writing Liberty data from a Power Aware analysis. You can reuse this database by using the vopt -pa_loadlibertydb argument.

- -pa_dumpupf <filename>

  (optional) Saves all UPF commands to an output file without any processing (even if the commands are not supported by Questa SIM). This is an alternate method of using the UPF command, save_upf, in uninterpreted mode.

- -pa_enable= <value>[+<value>]…

  (optional) Enables certain actions that Questa SIM performs during Power Aware simulation. You can use these values to control compatibility with previous releases.

  <value> — any of the following, in a plus (+) separated list.

  Refer to the section "Usage of vopt -pa_enable and -pa_disable" in the *Power Aware Simulatin User's Manual* for a complete list of values.

Note

To specify more than one value, use the plus (+) operator between values.
For example: vopt -pa_enable=insertls+insertiso+sourcesink

- -pa_excludefile <filename>

Case 1 — Using vopt -pa_prefix -pa_replacetop:

Skips Power Aware processing for a module, its instances in a particular hierarchical path, or signals in the design. The names and locations of these modules, instances, or signals are contained in an exclude file that directs Power Aware processing to skip all instantiation of <module_name> found within the instance <hier_path>.

<filename> — The name of a text file that specifies modules, instances, or signals you want to exclude from Power Aware verification. You can use the pound sign (#) to begin a comment line. Each entry in the file must be of the following form:

<module_name> [-a] <hier_path>

where

module_name can be any regular expression (enclosed in quotation marks)

-a is an optional switch that enables recursive exclusion of instances under module_name (that is, -a excludes the module module_name at path hier_path, plus ALL the instances of any module under module_name).

hier_path is the full pathname to the instance of the module you want to exclude. When a module instance is skipped, QuestaSim displays the following message:

```
** Note: (vopt-9691) Excluding power aware module '<module_name>'
in path '<hier_path>'.
```

When used with -a, hier_path limits the recursive exclusion to a particular scope.

Case 2 — Using vopt -pa_defertop:

As in Case 1, skips Power Aware processing for a module, its instances in a particular hierarchical path, or signals in the design by using an exclude file. The exclude file specifies instantiation of <module_name> found within the instance <hier_path> in the same way as in Case 1, with the following exception:

hier_path specifies the path to the instance of the module you want to exclude beginning at or below the instance of the pa_bbox dut_name (instead of using the full pathname)

- -pa_genrpt=[{nv | v} [{us | ud | u}] [b] [pa] [de] [cell] [conn]]

Note

To specify more than one report value, use the + operator between values.
For example: vopt -pa_genrpt=v+us+b

(optional) Generates the data for Power Aware report files. After simulation you will use the pa report command to write out the files to the current directory, or the directory specified with -pa_reportdir.

Use one of the following values to specify the level of information to include in the files:

nv — (non-verbose) Displays a count of the power aware elements in the design with respect to the power domains in the UPF file. This is the default if you do not specify nv or v.

v — (verbose) Displays the hierarchical path of individual power aware elements as well as their counts.

Use one of the following values to generate a report on the contents of the UPF file:

ud — Dynamic reporting. Includes time and polarity of controls (such as control port of a power switch, retention save and restore signals, isolation enable signal), plus Power Domain Status in the form of <Power Domain name>, <Strategy name>, <Control signal names>, <Active Sense>, <Current polarity value>. Dynamic UPF reporting is displayed in the transcript window.

us — Static reporting. Includes power domains, switches (with control ports and states), retention strategies (with save and restore controls), and isolation strategies (with isolation control and a list of isolated signals). Static UPF information is sent to a text file named *report.upf.txt*.

u — Static and dynamic.

Use any or all of the following values to generate more specific reports on the contents of the UPF file:

b — Provides bitwise expanded information for isolated and level-shifted ports in *report.upf.txt*.

pa — Provides information on Power Aware architecture (power domains, power switches, retention strategy, isolation strategy, level shifter strategy, PSTs) in *report.pa.txt*.

de — Provides information on power domain elements in *report.de.txt*.

Use either or both of the following values to generate reports about your design:

cell — Generates a report (*report.cell.txt*) about macro cells in your design.

conn — Generates a report (report.connection.txt) about connections between your UPF, HDL and Liberty files.

Refer to the section "UPF Reports" in the *Power Aware Simulation User's Manual* for more information.

- -pa_gls

  (optional) Enables gate-level simulation for a Power Aware analysis. Only required when running a purely gate-level simulation, not when running a mixed RTL and gate-level simulation.

- -pa_hiersep <alphanum_character>

  (optional) Specifies that the UPF uses a hierarchical path separator different from the default character, which is the slash (/) character.

- -pa_intcrptval0

  (optional) Sets the default corruption value to 0 (zero) instead of ‘left.

- -pa_lib <library_pathname>

  (optional) Specifies a different library location for Power Aware verification results than the current work directory. This is required if you want to keep results on the same design for different modes. You can use different libraries to preserve all sets of results, and then invoke vsim with the -pa_lib argument to specify which library location you want to use.

- -pa_libertyfiles=<liberty_pathname>[,<liberty_pathname> ...]

  (optional) Specifies a comma-separted list of locations of Liberty files. Refer to the section “Liberty Library Models” for more information.

- -pa_libertyrefresh=<database_pathname>

  (optional) Refreshes old library dumps in the cache directory created with vopt -pa_loadlibertydb or -pa_dumplibertydb.

- -pa_libertyupdate

  (optional) Uses dumps of specified Liberty file from current vopt run to replace existing dumps from previous runs in the database.

- -pa_loadlibertydb=<database_pathname>

  (optional) Specifies a location for loading a user-defined Liberty database created from a previous vopt run for a Power Aware analysis. This database will be used instead of the current Liberty library dump.

- -pa_lsthreshold <real>

  (optional) Sets a global threshold level for a Power Aware analysis containing multiple voltage levels, where <real> is any numerical value that specifies a voltage threshold.

  Use this argument when you know that level shifting is not required for particular range of voltage differences—you can then specify a global threshold. Otherwise, Questa SIM will flag missing level shifter errors even if the potential difference between two domains is within an acceptable range.

- -pa_maxpstcol <integer>

  (optional) Specifies the maximum number of columns shown in the PST table of the PST Analysis Report. Default is 8.

- -pa_modeltype=2

  (optional) Directs Questa SIM to ignore invalid transitions (such as X->1 or 0->X) on power-control signals for Power Aware simulation. This allows you to restrict Power Aware models to allow only valid transitions such as 0->1.

- -pa_noinpcorr

  Disables Liberty attribute-based input corruption, but does not affect Liberty attribute-based output corruption.

- -pa_nopartialontrans

  (optional) Disables the translation of PARTIAL_ON states of a supply net defined with the option -resolve parallel. Refer to the section set_partial_on_translation in the Power Aware User's Manual

- -pa_pgoutsynchchk

  Enables the check for synchronization of all inputs to a resolved parallel supply net that is connected to one or more supply sources that are internal pg pins of direction output.

- -pa_powertop <work_lib.power_module_name>

  (optional) Specifies the module in which the supplies referenced in UPF package functions supply_on/supply_off should be searched if the supplies are not found in the scope of calling function or -pa_top scope.

- -pa_prefix <hier_path>

  (optional) Specifies a hierarchical path to be prepended to path information in the UPF (or PCF) file. The paths in a UPF file start from a specific portion of the design, so you can use this argument to create a full path based on the top module on which you will run the vsim command.

- -pa_pstcompflags=[p | g | pg | pstids | useallps]

  (optional) Default is p. Controls the behavior of PST composition, which is enabled with -pa_enable=pstcomp.

  p — (default) Analyzes the domain-crossing pair: SourceSS.power and SinkSS.power.

  g — Analyzes the domain-crossing pair: SourceSS.ground and SinkSS.ground.

  pg — Analyzes one or both of the domain-crossing pairs: SourceSS.power and SinkSS.power and/or SourceSS.ground and SinkSS.ground.

  pstids — Add identifier tags to the PST tables of the PST section of the "PST Analysis Report"

  useallps — Analyzes all port states in the expansion of don't cares. By default don't cares in user-defined PSTs are expanded to only those port states that are used in another PST and not marked as unreachable.

  For arguments p, g, and pg refer to the section"Power State Tables"

  You can combine flags, using a plus sign (+), into any of the following:

```
-pa_pstcompflags=p+pstids
-pa_pstcompflags=g+pstids
-pa_pstcompflags=pg+pstids
```

- -pa_replacetop <string>

  (optional) Specifies a different name for the top-level module name and its instance name in the test bench (must be used with -pa_prefix). The value you specify for <string> replaces the first token specified in the paths defined in a UPF (or PCF) file before adding the pa_prefix <hier_path> to the paths. However, <string> will replace the first token only if the first token in the path is a top-level name that you included when invoking vopt. This is because control signals are allowed from a different hierarchy than the DUT itself.

- -pa_reportdir <pathname>

  (optional) Specifies an alternate directory for a Power Aware report file. The default location is the current directory.

- -pa_reportfile <filename>

  (optional) Specifies an alternate filename to the Power Aware report file. The default name is report.mspa.txt and it is saved in the current directory. Note that this argument does not create the Power Aware report file—you must specify -pa_genrpt to produce the report.

- -pa_rslvnetcheck

  (optional) Enables a check for when different states are driven by active drivers of a supply net defined with the -resolve parallel option. The check may issue warnings based on scenarios defined in the section set_partial_on_translation in the Power Aware User's Manual

- -pa_rtlinfo

  (optional) Run Power Aware verification in verbose mode. This helps identify register/latch modeling errors, such as non-synthesizable clocking styles.

- -pa_staticchecksonly

  (optional) Perform only power aware static checks of the design without performing the optimization and code generation tasks, resulting in a faster execution of the vopt command. This flow will still produce the static check reports that you can view with the pa report command. Once you have analyzed and verified the results, you can then run vopt without this option to proceed to a full vopt analysis and simulation.

  You can use this option along with -pa_checks and any "s" arguments to limit your static checks. For example, adding the following options to your standard vopt command will enable isolation static checking only:

  ```
  vopt <standard_options> -pa_staticchecksonly -pa_checks=si
  ```

- -pa_tclfile <filename>

  (optional) Use a Tcl file containing Unified Power Format (UPF) commands with save_upf -u (a UPF command). Note that save_upf -u is not defined by the standard for UPF 1.0, so

you must use the vopt -pa_tclfile argument to specify the output file when you use save_upf -u.

- -pa_top=<pathname>

  (optional) Specifies the hierarchy of a DUT to be used for Power Aware analysis of a UPF file. This allows analysis of the UPF file from a location other than the default top-of-design hierarchy used by vopt. For example, if you run vopt from the test bench location (tb) and the UPF scope is from the DUT (instantiated within the test bench as dut_inst), then you would specify the following:

  ```
  -pa_top /tb/dut_inst
  ```

- -pa_upf <filename>

  (optional) Specifies the name of a UPF file to be used by vopt.

- -pa_upfextensions=<extension>[+<extension>]

  (optional) Allows you to define and apply various UPF behaviors that are not supported by the current standard (UPF v2.0).

  For detailed information about the <extension> arguments, refer to the section "Supported UPF Extensions" in the *Power Aware Simulation User's Manual*.

  To specify more than one value, use the + operator between values. For example:

  vopt -pa_upfextensions=relatedsnet+genblk+v

  Specifying this argument with no values is the same as vopt -pa_upfextensions=default.

- -pa_upflist <file_list>

  (optional) Specfies a list of UPF files to vopt. Mutually exclusive with -pa_upf.

- -pa_upfoverloadsourcecmd

  (optional) Allows for the correct tracking of line numbers when you use the TCL source command to source a lower-level UPF file.

- -pa_upfsyntaxchecks

  (optional) Check only the syntax of UPF file specified by -pa_upf <filename>. Note that you must also use the -pa_upf argument with this argument.

- -pa_upfversion=[1.0 | 2.0 | 2.1]

  (optional) Specifies the version of UPF to use for power aware simulation.

  1.0 — UPF standard v1.0, originally developed by Accellera Organization, Inc.

  2.0 — (default) *IEEE Standard for the Design and Verification of Low Power Integrated Circuits* (IEEE Std1801-2009)

  2.0 — *IEEE Standard for the Design and Verification of Low Power Integrated Circuits* (IEEE Std1801-2013)

- -pa_zcorrupt

  (optional) Changes the default corruption value used at power-down from 'X' to 'Z'. This helps make debugging easier. Specifically, after using this argument, when Z represents the power-related corruption of a signal, corruption due to power shutoff can be distinguished from: X values that represent initial values, logical conflicts that arise in the design due to errors, or X values intentionally specified in the design.

  At power-up, these elements would take either the X value or their RTL value based on their logic types.) For sequential elements, if asynchronous controls are low at power up then the 'X' value is induced, otherwise the RTL(reset) value would prevail. For combinational elements, the RTL value would prevail.

  This change affects the following types:

  VHDL — std_logic, std_ulogic, and their aggregates.

  Verilog — all four state data types and their aggregates.

- -pdu

  Instructs vopt to preoptimize (black-box) a region of your design, allowing you to reuse the optimized portion and speed up future simulation and optimization runs. Refer to the section "Preoptimizing Regions of Your Design" for further information.

- -pdusavehierrefs[=<filename>]

  Creates a file with +acc settings. Required to maintain the necesssary visibilty of hierarchical references. By default the output filename is “mti_pdu_hier_refs” or you may specify a different filename with the optional <filename> argument. This argument is equivalent to the deprecated “-save_bbox_hier_refs” option that by default creates the output file “mti_bbox_hier_refs”).

- -pduspec{[+<instancePathName>] | [+[<libName>.]<moduleName>]}[+facc=<fileName>]

  (optional) Extracts visibility requirements (acc statements) for objects under a specified boundary into a file that is then used with vopt to create PDUs for the specified boundary modules or instances. Refer to Preserving Design Visibility with the Learn Flow and Preoptimizing Regions of Your Design for more information.

Note

Since the only intent of this step is the extraction of visibility requirements for the PDUs, vopt exits after the output files are written. Therefore, vopt does not require "-o" option when -pduspec is specified because no optimized output is created.

+facc — Specifies the file be saved in acc format.

<fileName> — A user specified string.

+<instancePathName> — A fully rooted path to an instance, for example: /top/dut1.

<libName>. — Specifies the library where the design has been compiled. Must be specified before <moduleName>.

+<moduleName>. — Specifies a module name.

- -permissive

  (optional) Allows messages in the LRM group of error messages to be downgraded to a warning. Also, it allows named port connections on bit-select and part-select ports, though only when multiple bit-select or part-select ports of same name are not present in the port list.

  You can produce a complete list by executing the command:

  ```
  verror -kind vopt -permissive
  ```

- -[no]primitiveaccess [=[-]<accesscodes>][+<objectselection>]

  (optional) Enable or disable access of PLI handles to primitive instances.

  (optional) Enable or disable line debugging.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

  <objectselection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

  <library> — Any Verilog, SystemVerilog, or VHDL library.

  <primary> — Specifies the design unit name.

  <secondary> — Specifies VHDL architecture.

  2. <instance_path> — A path to an instance.

  3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

- -[w]prof=<filename>

  (optional; -prof and -wprof are mutually exclusive) Enables CPU (-prof) or WALL (-wprof) time based profiling and saves the profile data to <filename>. Output from these arguments is used by Customer Support for debugging purposes.

- -proftick=<integer>

  (optional) Sets the time interval between the profile data collections. Default = 10.

- -pslext

  (optional) Enables PSL LTL and OBE operators. These operators are disabled by default.

- -pslfile_vh <filename>

  (optional) Identifies a VHDL PSL file to be read in during optimization. You can also specify a PSL file during compilation using the vcom command. Refer to "Applying PSL Assertions During Elaboration/Optimization" for more information.

- -pslfile_vl <filename>

  (optional) Identifies a Verilog PSL file to be read in during optimization. You can also specify a PSL file during compilation using the vlog command. Refer to"Applying PSL Assertions During Elaboration/Optimization" for more information.

- -quiet

  (optional) Disables 'Loading' messages.

- -[no]randmetastable [=[-]<accesscodes>][+<objectselection>]

  (optional) Enable or disable random insertion of 0 or 1 values on the output of a library cell when a timing violation is encountered.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

  <objectselection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

  <library> — Any Verilog, SystemVerilog, or VHDL library.

  <primary> — Specifies the design unit name.

  <secondary> — Specifies VHDL architecture.

  2. <instance_path> — A path to an instance.

  3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

- -source

  (optional) Displays the associated line of source code before each error message that is generated during compilation. By default, only the error message is displayed.

- -sc_arg <string> ...

  (optional) Specifies a string representing a startup argument which is subsequently accessible from within SystemC via the sc_argc() and sc_argv() functions (refer to "Accessing Command-Line Arguments in SystemC-22".

If multiple SystemC startup arguments are specified, each must have a separate -sc_arg argument. SystemC startup arguments returned via sc_argv() are in the order in which they appear on the command line. White space within the <string> will not be treated specially, and the string, white space and all, will be accessible as a single string among the strings returned by sc_argv().

- -sclib <library>

  (optional) Specifies the design library where the SystemC shared library is created. By default, the SystemC shared library is created in the logical work library. This argument is only necessary when the shared library is compiled in a design library other than the logical work directory. Refer to the sccom command for more information.

- -sdfmin | -sdftyp | -sdfmax[@<delayScale>] [<instance>=]<sdf_filename>

  (optional) Annotates VITAL or Verilog cells in the specified SDF file (a Standard Delay Format file) with minimum, typical, or maximum timing. Can also specify instances under VHDL generates as the SDF back-annotation point.

  @<delayScale> — scales all values by the specified value. For example, if you specify -sdfmax@1.5, all maximum values in the SDF file are scaled to 150% of their original value.

  Do not use this option if you scaled the SDF file while using the sdfcom command.

  <instance>= — specifies a specific instance for the associated SDF file. Use this when not performing backannotation at the top level.

  <sdf_filename> — specifies the file containing the SDF information.

- -sdfmaxerrors <n>

  (optional) Controls the number of Verilog SDF missing instance messages that will be emitted before terminating vsim. <n> is the maximum number of missing instance error messages to be emitted. The default number is 5.

- -showsubprograms | -noshowsubprograms

  (optional) Toggles viewing VHDL subprogram scopes on the command line and in GUI windows, for example, the Structure window. The default is not to show subprogram scopes.

- +staticchecks[=<args>]

  (optional) Performs a series of static checks on VHDL and Verilog designs, where <args> is a list of arguments. The command performs all checks if you do not specify any arguments to this switch.

  Produces a file (*static_checks.txt*) in the current run directory containing messages about any static check violations. You can rename the file with the -staticchecksfile switch.

  r — checks for race conditions, either write-write races (multiple drivers) or read-write races. Resulting messages have the label: STATIC_RACE_CHECK.

  These checks are only on a a per-module basis.

To enable multiple driver checks for VHDL you must also specify the -staticchecksmdvhdl switch. This is because the multiple driver may result from a resolution function.

c — checks for simulation-synthesis mismatches related to full/parallel case pragmas. Resulting messages have the label: STATIC_CASE_CHECK.

s — checks for simulation-synthesis mismatches related to sensitivity lists; missing objects, duplicate elements, and redundant elements. Resulting messages have the label: STATIC_SENSLIST_CHECK.

f — checks for simulation-synthesis mismatches related to subprograms (functions). Resulting messages have the label: STATIC_FUNCTION_CHECK.

x — checks for simulation-synthesis mismatches related to reading of four-state values (X, Z, U, or W). Resulting messages have the label: STATIC_XZUW_CHECK.

This check does not issue any warnings for casex reading "x" or "z", or casez reading "z".

d — checks for non-synthesizable constructs; untested edge triggers, implicit state machine with different clocks or where the first statement is not within event control, output driven by multiple clocks, improper mixing of control signals, blocking and non-blocking assignments for the same signal, and asynchronous loading. Resulting messages have the label: STATIC_SYNTH_CHECK.

Usage examples:

```
+staticchecks           #Executes all checks
+staticchecks=r         #Executes only the check for race conditions
+staticchecks=rcs       #Executes three of the checks
```

- -staticchecksfile <filename>

  (optional) Redirects the output of the +staticchecks switch to <filename>. By default, this file is named *static_checks.txt*.

- -staticchecksmdvhdl

  (optional) When used with +staticchecks=r, enables the check on multiple driver race conditions for VHDL designs. By default, this check is deactivated because the multiple driver for VHDL could be related to a resolution function.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance will take effect.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+)

character. Setting this switch will add or subtract features and modes from the settings in the Stats *modelsim.ini* variable.

Features

all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

cmd — (default) Echo the command line.

msg — (default) Display error and warning summary at the end of command execution.

none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

perf — Display time and memory performance statistics.

time — (default) Display Start, End, and Elapsed times.

Modes

Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vopt -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vopt -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

kb — Print performance statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

Note

vopt -quiet disables all default or user-specified -stats features.

- -suppress <msg_number>[,<msg_number>,...]

  (optional) Prevents the specified message(s) from displaying. You cannot suppress Fatal or Internal messages. Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -svext=[+|-]<extension>[,[+|-]<extension>]…

  (optional) Enables SystemVerilog language extensions through a comma-separated list of arguments.

  [+ | -] — controls activation of the *extension*. Remember that arguments on the command line control the precedence; any settings to this switch will override your settings of the SvExtensions *modelsim.ini* variable.

  + — activates the *extension*.

  - — deactivates the *extension*.

If you do not specify either a “+” or “-”, the command assumes you are activating the specified *extension*.

<extension> —

acum — Specifies that the get(), try_get(), peek(), and try_peek() methods on an untyped mailbox will return successfully if the argument passed is assignment-compatible with the entry in the mailbox. The LRM-compliant behavior is to return successfully only if the argument and entry are of equivalent types.

atpi — Use type names as port identifiers. Disabled when compiling with -pedanticerrors.

catx — Allow an assignment of a single unsized constant in a concat to be treated as an assignment of 'default:val'.

daoa — Allows the passing a dynamic array as the actual argument of DPI open array output port. Without this option, a runtime error, similar to the following, is generated, which is compliant with LRM requirement.

```
# ** Fatal: (vsim-2211) A dynamic array cannot be passed as an
argument to the DPI import function 'impcall' because the formal 'o'
is an unsized output.
#    Time: 0 ns  Iteration: 0  Process: /top/#INITIAL#56 File:
dynarray.sv
# Fatal error in Module dynarray_sv_unit at dynarray.sv line 2
```

evis — Supports the expansion of environment variables within `include path names. For example, if MYPATH exists in the environment then it will be expanded in the following:

```
`include "$MYPATH/inc.svh"
```

feci — Treat constant expressions in a foreach loop variable index as constant.

fin0 — Treats $finish() system call as $finish(0), which results in no diagnostic information being printed.

idcl — Allows passing of import DPI call locations as implicit scopes.

iddp — Ignore the DPI task disable protocol check.

pae — Automatically export all symbols imported and referenced in a package.

sccts — Process string concatenations converting the result to string type.

spsl — (default) Search for packages in source libraries specified with -y and +libext.

stop0 — Treats $stop and $stop() as $stop(0), which results in no diagnostic information being printed.

udm0 — Expands any undefined macro with the text “1'b0”.

uslt — (default) Promote unused design units found in source library files specified with the -y option to top-level design units.

Multiple extensions are specified as a comma separated list. For example:

vlog -svext=+feci,-uslt,pae

- -[no]systfoverride [=[-]<accesscodes>][+<objectselection>]

  (optional) Enable or disable user-specified overrides of built-in system tasks (such as $write or $display).

  NOTE: In order for specified overrides to take effect in certain design regions, you must also ensure that vopt -systfoverride is in effect. If it is not in effect, the system-provided system task will remain in effect, even if you have made an overridden version available with PLI.

  =<accesscodes> — optionally, one or more of the following characters. If <accesscodes> is omitted, the entire set of access specifiers is applied. <spec> can contain a non-negative integer to specify a recursive level downwards from the specified object.

  c — Preserve connectivity access on selected object.

  r — Preserve read access on selected object.

  w — Preserve write access on selected object.

  <objectselection> — Can take one of the following forms:

  1. [<library>.]<primary>[.(<secondary>)] — Specifies a library name together with a design unit name.

  <library> — Any Verilog, SystemVerilog, or VHDL library.

  <primary> — Specifies the design unit name.

  <secondary> — Specifies VHDL architecture.

  2. <instance_path> — A path to an instance.

  3. <design_unit>/<instance> — Specifies an instance in a specific design unit.

- -tab <tabfile>

  (optional) Specifies the location of a Synopsys VCS table file (.tab), which vopt uses to improve the visibility of PLI functions in the design.

  <tabfile> — The location of a *.tab* file containing information about PLI functions. The tool expects the *.tab* file to be based on Synopsys VCS version 7.2 syntax. Because the format for this file is non-standard, changes to the format are outside of the control of Mentor Graphics.

  When you use the Three-step optimization flow, you must specify this switch on both the vopt and vsim command lines. This is because vopt uses this file to improve the visibility of PLI functions and vsim uses it to register the PLI functions.

- -tbxhierrefs[[+<instancePathName>] | [+[<libName>.]<moduleName>]] [+f=<fileName>]

  (optional) Extracts module, filename and line number information for HDL hierarchical references that cross into a specified module or instance boundary. Optionally writes the information into an output file specified by the +f argument. You must manually edit the flagged lines if you wish to disable such hierarchical access.

Note

Since the only intent of this step is the extraction of visibility requirements, vopt exits after the output files are written. Therefore, vopt does not require the "-o" option when -tbxhierrefs is specified because no optimized output is created.

+f=<fileName> — A user specified string.

+<instance PathName> — A fully rooted path to an instance, for example: /top/dut1.

<libName>. — Specifies the library where the design has been compiled. Must be specified before <moduleName>.

+<moduleName> — Specifies a module.

- -timescale[=]|[ ]<time_units>/<time_precision>

  (optional) Specifies the default timescale for all design unit types (modules, interfaces, programs, packages, checkers, and so forth) not having an explicit timescale directive in effect during compilation. Only affects design units that have not had a timescale assignment made previously in HDL source or with vlog-timescale.

  The format of the -timescale argument is the same as that of the `timescale directive. An equal sign (=) or whitespace is accepted between option and arguments in which case <time_units / <time_precision must be enclosed in quotation marks ("). The format for <*time_units*> and <*time_precision*> is <*n*><*units*>. The value of <*n*> must be 1, 10, or 100. The value of <*units*> must be fs, ps, ns, us, ms, or s. In addition, the <*time_precision*> must be smaller than or equal to the <*time_units*>. Refer to "Simulator Resolution Limit (Verilog)" for more information.

- -togglecountlimit <int>

  (optional) Limits the toggle coverage count, <int>, for a toggle node. After the limit is reached, further activity on the node is ignored for toggle coverage. All possible transition edges must reach this count for the limit to take effect. For example, if you are collecting toggle data on 0->1 and 1->0 transitions, both transition counts must reach the limit. If you are collecting "full" data on 6 edge transitions, all 6 must reach the limit. Overrides the global value set by the ToggleCountLimit *modelsim.ini* variable.

- -togglepportsonly

  (optional) Enables only ports for inclusion in toggle coverage numbers; no internal signals are not included in coverage metrics. If you want to see coverage of all ports in the design, use the "vopt +acc=p" command — but note that there could be a significant performance penalty due to the fact that inlining is turned off. You can selectively enable toggle coverage on specific ports by using the "vopt +acc=p+<selection>" command. Overrides any global value set by the TogglePortsOnly *modelsim.ini* variable.

- -togglewidthlimit <int>

  (optional) Sets the maximum width of signals, <int>, that are automatically added to toggle coverage with the -cover t argument. Can be set on design unit basis. Overrides the global value of the ToggleWidthLimit *modelsim.ini* variable.

- +typdelays

  (default) Selects typical delays from the "min:typ:max" expressions. If preferred, you can defer delay selection until simulation time by using vsim -sdfmin.

  If you specify the +mindelays, +typdelays, or +maxdelays flag with vopt, and specify a different flag with vsim, the simulation will be able to use the delay value based upon the flag specified with vopt. You must specify vsim -novopt to force the simulator to use the delay flag specified with vsim.

- -undefsyms={<args>}

  (optional) Manages the undefined symbols in the shared libraries to be loaded into the simulator. You can also manage undefined symbols with the UndefSyms *modelsim.ini* variable.

  {<args>}

  You must specify at least one argument.

  on — Enables automatic generation of stub definitions for undefined symbols and permits loading of the shared libraries despite the undefined symbols.

  off — (default) Disables loading of undefined symbols. Undefined symbols trigger an immediate shared library loading failure.

  verbose — Permits loading to the shared libraries despite the undefined symbols and reports the undefined symbols for each shared library.

- -version

  (optional) Returns the version of the optimizer as used by the licensing tools.

- -warning <msg_number>[,<msg_number>,...]

  (optional) Changes the severity level of the specified message(s) to "warning." Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -warning error

  (optional) Reports all warnings as errors.

- -work <library_name>

  (optional) Specifies a logical name or pathname of a library that is to be mapped to the logical library work. By default, the optimized output for the design is added to the work library. The specified pathname overrides the pathname specified for work in the project file.

- -xprop[,mode=<modecode>][,report=<severitylevel>][,object=<objectselection>]

  (optional) Defines the level of pessimism for X propagation for specified modules and/or instances. The effect of this argument can be overridden with the xprop disable command.

  <modecode> = [resolve | pass | xtrap]

resolve (default) —all possible branches are evaluated: if the output values from all branches are the same, the output is driven to that known value. If the output values are not the same, the output is driven to ‘X’. Simulation is affected, and can fail.

pass — if any condition goes ‘X’, an assertion message is output and all writers go ‘X’. Simulation is affected, and can fail.

xtrap — if any condition goes ‘X’, only assertion messages are output; simulation behavior is unaffected.

<severitylevel> = [fatal|error|warning|info|suppress] — operates only on the severity level specified, error is default.

<objectselection> = <full/path/instance> | <du>[/<inst>] | <hierlevel>

‘+’ is used to separate the different object selections.
‘.’ is used to indicate recursion.
‘-’ is used as a negation modifier to exclude modules and instances.
Regexp are supported for module searching. For example:

vopt -xprop,object=/top/u1+/top/u2+/top/u3.-top/u3.inst1+du1.

This example includes: u1, u2, all of u3 recursively (exclusive of inst1), and all of du1 recursively.

For details on X propagation, refer to “X Propagation in Simulation”.

- <design_unit>

  One or more top-level design units that you want to optimize. Required. Package names may be used, and will be treated as a top-level design unit.

- -o <name>

  Specifies a name for the optimized version of the design. Required. The name can contain lower- and upper-case alpha characters, numeric characters or underscores (_). This vopt option is not permitted when using vsim -voptargs="<args>" command.

Examples

- Run optimizations on top-level design unit *top* and produce an optimized design unit named "mydesign". The simulator vsim is then invoked on design unit *mydesign*.

```
vopt top -o mydesign
vsim mydesign
```

- Run optimizations both top-level design units *top* and *testtop* and produce a global optimized design unit named mydesign.

```
vopt top testtop -o mydesign
```

- Run optimizations on top-level design unit *top* but preserve all visibility. Names the optimized design "mydesign."

```
vopt top +acc -o mydesign
```

- Run optimizations on top-level design unit *top* but preserve visibility on sub-module *foo*. Names the optimized design "mydesign."

```
vopt top +acc+foo -o mydesign
```

- Run optimizations on top-level design unit *top* but preserve visibility on sub-module *foo* and all its children.

```
vopt top +acc+foo. -o mydesign
```

- Run optimizations on top-level design unit *top* but enable net and register access in all modules in the design. Names the optimized design "mydesign."

```
vopt top +acc=rn -o mydesign
```

- The -fsmmultitrans option enables detection and reporting of multi-state transitions when used with the +cover f argument.

```
vopt -o opt_ttop top +cover=f -fsmmultitrans
```

- Run Power Aware using -pa_prefix and -pa_replacetop while specifying an exclude file (Case 1—exclude file must specify full hierarchy of excluded instance:

```
vopt -pa_prefix "/interleaver_tester/" -pa_replacetop
"interleaver1"  -o my_chip -pa_excludefile bar_exclude

#Contents of bar_exclude:
foo /interleaver_tester/interleaver1/foo[1]/bar -s ctl
```

- Run Power Aware using -pa_defertop while specifying an exclude file (Case 2—exclude file needs to specify hierarchy of excluded instance only beginning at dut_name of pa_bbox instance):

```
vopt -pa_defertop "interleaver"  -o my_chip -pa_excludefile
bar_exclude

#Contents of bar_exclude:
foo foo[1]/bar -s ctl
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vopt -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all *modelsim.ini* settings and then enables the perf option.

```
vopt -stats=time,cmd,msg -stats=none,perf
```

- Specify access visibility for a VHDL design unit named myarch:

```
vopt -floatgenerics+myentity(myarch)
```

- Specify access visibility for a Verilog design unit named Demux:

```
vopt -bitscalar+Demux
```

- Specify access visibility for a VHDL design unit (mux) within the work library:

```
vopt -primitiveaccess+work.mux
```

- Specify access visibility for a VHDL instance (r) within a design unit (dut):

```
vopt -linedebug+dut/r
```

- Specify access visibility for a VHDL instance (u1) and below:

```
vopt -assertdebug+/top/u1.
```

- Specify access visibility for a VHDL object (myreg):

```
vopt -randmetastable+/top/myreg
```

- Specify access visibility for a VHDL region and below:

```
vopt -fsmdebug+'/top/gen(3).'
```

- Specify access visibility for a VHDL region and two levels below:

```
vopt -nobitscalars+'/top/gen(3)+2.'
```

Related Topics

- "Optimizing Designs with vopt"
- "Preoptimizing Regions of Your Design"

vsim

The vsim command invokes the VSIM simulator, which you can use to view the results of a previous simulation run (when invoked with the -view switch)

You can also view coverage data stored in a UCDB or a coverstore from a previous simulation run (when invoked with the -viewcov switch). A coverstore is an auto-saved coverage location, specified by the vsim -coverstore command.

You can simulate a VHDL configuration or an entity/architecture pair, a Verilog module or configuration, a SystemC module, or an optimized design.

If you specify a VHDL configuration, it is invalid to specify an architecture. During elaboration, Questa SIM determines if the source has been modified since the last compile.

To manually interrupt design loading, use the Break key or press <Ctrl-C> from a shell.

You can invoke vsim from a command prompt or in the Transcript window of the Main window. You can also invoke it from the GUI by selecting Simulate > Start Simulation.

Package names may be used at the command line and will be treated as top-level design units.

All arguments to the vsim command are case-sensitive; for example, -g and -G are not equivalent.

Syntax

This section (Syntax) lists all vsim arguments in alphabetical order.

The Arguments section groups the argument descriptions into the following categories:

- All languages
- VHDL Arguments
- Verilog Arguments
- SystemC Arguments
- Visualizer Debug Environment Arguments
- Object Arguments

vsim [options]

[options]:

[-32 | -64]

[-absentisempty] [-accessobjdebug | -noaccessobjdebug] [-assertcover | -noassertcover] [-assertdebug | -noassertdebug] [+alt_path_delays] [-assertfile <filename>] [-assume | -noassume] [-autoexclusionsdisable=<exclusion_type>] [-autoprofile[=<profile_database>]]

[-batch] [+bitblast[=[iopath | tcheck]]]

[-c] [-capacity] [-checkvifacedrivers] [-classdebug | -noclassdebug]
[+class_memleak_threshold+<n>]
[-codelink=<home_path> | -nocodelink] [-colormap new] [-compress_elab]
[-coverage][-coveranalysis] [-covercountnone] [-coverenhanced]
([-coverstore <dir_path> -testname <name> [-multicount[=-a|b|c|-d|f|-g|s|t]]])
[-cppinstall <[gcc | g++] version>] [-cpppath <filename>]
[-cvgbintstamp] [-cvgcollapseembeddedinstances] [-cvghaltillbin]
[-cvgmaxrptrhscross] [-cvgmergeinstances | -nocvgmergeinstances]
[-cvgopt[=[+|-]<mode>[,[+|-]<mode>]...]] [-cvgperinstance]
[-cvgprecollect <ucdb_file>] [-cvgsingledefaultbin]
[-cvgzwnocollect]

[-debugdb=<db_pathname>] [-default_radix <radix>] [-defaultstdlogicinittoz]
[+delayed_timing_checks] [-display <display_spec>]
[-displaymsgmode both | tran | wlf] [-do “<command_string>” | <do_file_name>]
[-donotcollapsepartiallydriven] [-dpicppinstall <[gcc | g++] version>]
[-dpicpppath <pathname>] [-dpiforceheader] [-dpiheader] [-dpilib <libname>]
[-dpioutoftheblue 0 | 1 | 2] [+dumpports+collapse | +dumpports+nocollapse]
[+dumpports+direction] [+dumpports+no_strength_range] [+dumpports+unique]

[-error <msg_number>[,<msg_number>,…]] [-elab <filename>]
[-elab_cont <filename>] [-elab_defer_fli] [-enumfirstinit]
[-extendedtogglemode 1|2|3]

[-f <filename>] [-fatal <msg_number>[,<msg_number>,…]] [-feccountlimit [<n> | 0]]
[-filemap_elab <HDLfilename>=<NEWfilename>] [-foreign <attribute>]
[-fsmdebug]

[-g <Name>=<Value> …] [-G<Name>=<Value> …] [-gblso
<shared_obj>[,<shared_obj>]]
[-gconrun | -nogconrun] [-gconstep | -nogconstep] [-gcthreshold <n>]
[-geometry <geometry_spec>] [-gui]

[-hazards] [-help]

[-i] [-ignoreinilibs] [-immedassert | -noimmedassert] [+initmem+<seed>]
[+initreg+<seed>] [-initreport <filename>] [+initregNBA | +noinitregNBA]
[-installcolormap]

[-keeploaded] [-keeploadedrestart] [-keepstdout]

[-logfile <filename> | -l <filename> | -nolog] [-L <library_name> …]
[-ldflags <"linkopts">] [-lib <libname>] [<library_name>.<design_unit>]
[-learn <root_file_name>] [<license_option>]
[-Lf <library_name> …] [-L mtiPA] [-load_elab <filename>]

[+max_bits_per_wave+<n>] [+maxdelays] [+mindelays] [-memprof] [-memprof+call]
[-memprof+file=<filename>]
[-memprof+fileonly=<filename>] [-mlopt] [-modelsimini <path/modelsim.ini>]
[--msgfile <filename>] [-msglimit [all, | none,] <msgNumber>[, <msgNumber>,…]]
[-msglimitcount <limit_value> -msglimit [all, | none,] <msgNumber>[,
<msgNumber>...]] [-msgmode both | tran | wlf]

[-multisource_delay min | max | latest] [+multisource_int_delays] [-mvchome <path>]

[-name <name>] [-noappendclose] [+no_autodtc] [-noautoldlibpath] [-nodpiexports] [-no_autoacc] [+no_cancelled_e_msg] [+no_glitch_msg] [+no_neg_tchk] [+no_notifier] [+no_path_edge] [+no_pulse_msg] [-no_risefall_delaynets] [+no_show_cancelled_e] [+no_tchk_msg] [-nocollapse] [-nocapacity] [-nocompress] [-nocrossautobins[=[cond|uncond]]] [-nocvg] [-noexcludehiz] [-noexcludeternary] [-nofileshare] [-noimmedca] [-noimplicitcoverpoint] [-noglitch] [-noscmainscopename] [+nosdferror] [+nosdfwarn] [+nospecify] [-nostdout] [-note <msg_number>[,<msg_number>,…]] [+notifier_ondetect] [+notiftoggle01[+<seed>]] [+notimingchecks | +ntcnotchks] [-notogglealias] [-novhdlvariablelogging] [+nowarnBSOB] [+nowarn<CODE | number>] [-nowiremodelforce] [+ntc_warn] [+ntcnotchks]

[-oldvhdlforgennames] [-onfinish ask | stop | exit | final] [-optionset <optionset_name>] [-OVMdebug] [-ovmtestname] [+OVM_TESTNAME=<testname>]

[-pa] [-pa_allowtimezeroevent[=all]] [-pa_debugdir <directory>] [-pa_disabletimezeroevent] [-pa_gls <testbench_top>] [-pa_highlight] [-pa_lib <pathname>] [-pa_togglelimit=<n>] [-pa_top <pathname>] [-pduignore[=<instpath>]] [-pedanticerrors] [-permissive] [-permit_unmatched_virtual_intf] [-pli "<object list>"] [-plicompatdefault [latest | 2009 | 2005 | 2001]] [+<plusarg>] [-postsimdataflow] [-printsimstats[=[<val>][v]]] [-psl | -nopsl] [-psloneattempt] [-pslinfinitethreshold=<integer>] [+pulse_e/<percent>] [+pulse_e_style_ondetect] [+pulse_e_style_onevent] [+pulse_r/<percent>]

[-quiet] [-qwavedb[=+<option>[+<option>+...]] [+qwavedb_include_cells] [+qwavedb_log_vhdl_vars] [+qwavedb_dump_start+<start_time>] [+qwavedb_dump_stop+<stop_time>]

[-[no]randmetastable [+<seed>]] [-restore <filename>] [-runinit]

[-sc_arg <string> ...] [-scdpidebug] [-sclib <library>] [-scstacksize <value>] [+sdf_iopath_to_prim_ok] [+sdf_nocheck_celltype] [-sdfmin | -sdftyp | -sdfmax[@<delayScale>] [<instance>=]<sdf_filename>] [-sdfminr | -sdftypr | -sdfmaxr[@<delayScale>] [<instance>=]<sdf_filename>] [-sdfmaxerrors <n>] [-sdfnoerror] [-sdfnowarn] [-sdfreport=<filename>] [+sdf_report_unannotated_insts] [+sdf_verbose] [-showlibsearchpath] [-stackcheck] [-std_input <filename>] [-std_output <filename>] [+show_cancelled_e] [-solvebeforeerrorseverity=<value>] [-solveengine {auto | bdd | act}] [-solvefaildebug=<value>] [-solvefailtestcase[=<filename>]] [-solveprofile] [-solverev <version>] [-solvetimeout <val>] [-solveverbose] [-stats [=[+ | -]<feature>[,[+ | -]<mode>]] [-strictvital] [-suppress <msg_number>[,<msg_number>,…]] [-sv_lib <shared_obj>] [-sv_liblist <filename>] [-sv_root <dirname>] [-sv_seed <integer> | random] [-svext=[+|-]<extension>[,[+|-]<extension>]…] [-sva | -nosva] [-sync] [-syncio | -nosyncio]

[-t [<multiplier>]<time_unit>] [-tab <tabfile>] [-tag <string>] [-title <title>]
[-togglecountlimit <int>] [-togglefixedsizearray | -notogglefixedsizearray]
[-togglemaxfixedsizearray <int>] [-togglemaxintvalues <int>]
[-togglemaxrealvalues <int>] [-togglepackedasvec] [-togglepportsonly]
[-togglevhdlrecords | -notogglevhdlrecords]
[-togglevlogenumbits] [-notoggleints] [-togglevlogints | -notogglevlogints]
[-togglevlogreal | -notogglevlogreal] [-togglewidthlimit <int>] [-trace_dpi <val>]
[-trace_foreign <int>] [+transport_int_delays]
[+transport_path_delays] [+typdelays]

[-ucdbteststatusmsgfilter <TCL_subtext>] [-udpcountlimit [<n> | 0]]
[-undefsyms={<args>}] [-usenonstdcoveragesavesysf] [-uvmcontrol={<args>}]
[-uvmtestname] [+UVM_TESTNAME=<testname>]

[-v2k_int_delays] [-vcdread <filename>] [-vcdstim [<instance>=]<filename>]
[-version] [-vhdlmergepdupackage] [-vhdlseparatepdupackage]
[-vhdlvariablelogging] [-view [<alias_name>=]<WLF_filename>]
[-viewcov [<dataset_name>=]<UCDB_filename>] [-visual <visual>]
[-vital2.2b] [+vlog_retain_on | +vlog_retain_off]
[+vlog_retain_same2same_on | +vlog_retain_same2same_off]
[-vopt | -novopt] [-voptargs="<args>"] [-vopt_verbose]

[-warning <msg_number>[,<msg_number>,…]] [-warning error] [-wlf <file_name>]
[-wlfcachesize <n>] [-wlfcollapsedelta] [-wlfcollapsetime] [-nowlfcollapse]
[-wlfcompress] [-nowlfcompress] [-wlfdeleteonquit] [-nowlfdeleteonquit]
[-wlflock] [-nowlflock] [-wlfopt] [-nowlfopt] [-wlfsimcachesize <n>]
[-wlfslim <size>] [-wlftlim <duration>] [-wrealdefaultzero]
[-wreal_resolution <val>]

Arguments

All languages

- -32 | -64

  (optional) Specifies whether vsim uses the 32-bit or 64-bit executable, where 32-bit is the default.

  These options apply only to the supported Linux operating systems.

  These options override the MTI_VCO_MODE environment variable, which applies only to executables used from the *<install_dir>/bin/* directory. Therefore, these options are ignored if you run vsim from an *<install_dir>/<platform>/* directory.

  You can only specify these options on the command line, therefore they are not recognized as part of a file used with the -f switch.

- -assertcover

  (optional) Enables extended count information including: pass count, vacuous pass count, disabled count, attempted count. These enabled counts are passed to coverage reports during both live simulation (coverage report) and post-processing (vcover report) modes. The default behavior can be changed by setting the AssertionCover variable in the *modelsim.ini* file.

The details involving the assertion counts reported:

- At simulation time, if "-assertcover" is present, then Assertion Passes are counted and Assertion Successes are those assertions that never failed and passed at least once.
- In the absence of "-assertcover", Assertion Passes are not counted, and Assertion Successes are those assertions that never failed.
- Assertion Attempts are the sum of all evaluated assertions.

- -assertdebug

  (optional) If assertions are logged, this option stores the assertion data in the WLF file so assertion passes and failures can be analyzed in the assertion debug pane of the Wave window. It also enables:

  - Action settings for assertion passes, starts, and antecedent matches.
  - Assertion thread viewing (ATV) feature.
  - Failed expression analysis.
  - Extended count information including: pass count, vacuous pass count, disabled count, attempted count, active thread count, peak active thread count. However, if you only want this information, you should specify -assertcover instead.
  - Causality traceback feature.
  - Reporting of assertion pass messages for PSL only (disabled by default).

  The default behavior can be changed by setting the AssertionDebug variable in the *modelsim.ini* file. If no debuggable assertions are found in the design, a warning is issued. Refer to "Analyzing Assertion Failures in the Assertion Debug Pane of the Wave Window" for more info.

  When this argument is used, "100% coverage" is achieved if an assertion passes non-vacuously and never failed. Without -assertdebug set, "100% coverage" is achieved if the assertion never failed.

- -assertfile <filename>

  (optional) (optional) Designates an alternative file for recording VHDL/PSL/SVA assertion messages.

  An alternate file may also be specified by the AssertFile *modelsim.ini* variable. By default, assertion messages are output to the file specified by the TranscriptFile variable in the *modelsim.ini* file. Refer to "Creating a Transcript File" for more information.

- -assume

  (optional) Simulates PSL and SystemVerilog assume directives as though they were assert directives. This option may also be specified with the SimulateAssumeDirectives variable in the *modelsim.ini* file. Refer to "Processing Assume Directives" for more information.

- -autoexclusionsdisable=<exclusion_type>

  (optional) Disables automatic code coverage exclusions for:

  - FSMs and it's transitions
  - VHDL and SystemVerilog immediate and concurrent assertions and their action blocks.

  <exclusion_type> — A comma-separated list of values that specify the automatic exclusions you wish to disable, where the values are:

  fsm — disables automatic exclusion of FSMs

  assertions — disables automatic exclusion of VHDL and SystemVerilog immediate and concurrent assertions.

  none — equivalent to "fsm,assertions"

  To change this default behavior, use the AutoExclusionsDisable variable in the *modelsim.ini* file. If an FSM state is excluded, then all transitions from and to this state are also excluded.

- -autoprofile[=<profile_database>]

  (optional) Use this option to cause Vsim to automatically collect performance profile data without requiring the use of profile commands. (Does not collect memory profile data.) You can specify a profile database name (optional):

  [=<profile_database>] — If no <profile_database> is specified, then a name using date, time and process PID is created, otherwise the supplied profile database name is used.

Note

The following profile commands are disabled when -autoprofile is used:
profile interval
profile clear
profile on
profile off
profile reload
profile open
profile option keep_unknown

See the individual profile commands reference entries and "Profiling Performance and Memory Use" in the *Questa SIM User's Manual* for more information.

- +autofindloop

  (optional) Enables detection of infinite zero-delay loops when the IterationLimit is reached. Refer to the section "Detecting Infinite Zero-Delay Loops" in the User's Manual for more information.

- -batch

  (optional) Runs scripted batch simulations via the -do argument to vsim. Must be specified from a Windows command prompt or a UNIX terminal. The simulator returns an error if

-batch is used with the -c, the -gui, or the -i argument to vsim. You can edit the BatchMode modelsim.ini variable to automatically run in batch mode when none of -c, -gui, or -i are used.

By default, vsim -batch prevents automatic creation of a transcript file by disabling the TranscriptFile *modelsim.ini* variable and sending transcript data to stdout. You can create a transcript file by specifying the -logfile <filename> argument to vsim or by uncommenting the BatchTranscriptFile *modelsim.ini* variable. You can also disable sending transcript data to stdout by specifying vsim -nostdout however, you must then save transcript data to a file. Refer to "Batch Mode" for more information about saving transcript data.

- +bitblast[=[iopath | tcheck]]

  (optional) Enables bit-blasting of specify block iopaths and timing checks (tchecks) with wide atomic ports. Without the optional qualifiers, the switch operates on both specify paths and tchecks. The qualifiers work as follows:

  +bitblast=iopath — bit-blasts only specify paths with wide ports.

  +bitblast=tcheck — bit-blasts only tchecks with wide ports.

  This switch is intended for use with applications employing SDF annotation.

- -c

  (optional) Specifies that the simulator is to be run in command-line mode. Refer to "Modes of Operation" for more information.

- -capacity

  (optional) Enables the fine-grain analysis display of memory capacity where the default is coarse-grain analysis display.

- -codelink=<home_path> | -nocodelink

  (optional) Specifies the home directory pathname for the CodeLink product. This argument will override the current CODELINK_HOME environment variable value, or you can use it instead of setting the environment variable.

  Use the -nocodelink argument to disable loading of the CodeLink user interface during the simulation session.

  These arguments can only be used when first invoking the vsim command and not later from a .do script.

Note

The -codelink= switch cannot be used at the same time as explicit -foreign codelink options. Only one or the other should be used, never both at the same time.

- -colormap new

  (optional) Specifies that the window should have a new private colormap instead of using the default colormap for the screen.

- -compress_elab

  (optional) Compresses an elaboration file when it is created. Refer to "Simulating with an Elaboration File" for more information.

- -coverage

  (optional) Enables code coverage statistics collection during simulation.

  Important: in order for coverage to be collected and displayed, you must have used +cover options during compilation or optimization.

- -coveranalysis

  (optional) Display alias toggle nodes in coverage reports and in the GUI, and save those nodes in the UCDB files.

- -covercountnone

  (optional) This switch applies to condition and expression coverage UDP tables.

  Thus, it has no effect unless UDP is enabled for coverage with vcom/vlog/vopt -coverudp.

  If this option is used and a match occurs in more than one row in the UDP table, none of the counts for all matching rows is incremented. By default, counts are incremented for all matching rows. You can change the default behavior by editing the CoverCountAll variable in the *modelsim.ini* file. Please refer to the "Code Coverage" chapter in the User's Manual for more information.

- -coverenhanced

  (optional) Enables non-critical functionality which might change the appearance or content of coverage metrics. This argument only has an effect in letter releases (10.0a, 10.0b, and so on). In major releases (10.0, 10.1, and so on), all coverage enhancements present in previous letter release streams are enabled by default, and -coverenhanced is no longer necessary to enable these enhancements. Bug fixes important to the correctness of coverage numbers are always enabled by default, with no need for -coverenhanced. Since the exact nature of -coverenhanced varies from release to release, the details of the enhancements it enables are present in the product release notes rather than in the Command Reference. For these details, search the release notes using the string "coverenhanced".

- -coverstore <dir_path> -testname <name> [-multicount[=-a|b|c|-d|f|-g|s|t]]

  (required to create a coverstore for a coverstore merge) Saves the coverage data at the end of simulation, without need for coverage save command. Requires also the specification of -testname <name>, which is the name of the test whose coverage is being saved. This mode of saving coverage is most useful where code coverage remains unchanged for different tests in a regression run. The result is a faster merge. All simulation runs in the regression which use -coverstore must use the same directory path, and their design hierarchy must be identical.

  -multicount[=-a|b|c|-d|f|-g|s|t]

  (optional) Flips the coverage types to single-bit from multi-bit, or to multi-bit from single-bit. Functional coverage types (assertions, cover directives, and covegroups)

are multi-bit by default, and are so indicated by the prepended '-' . You can switch them to single-bit using the -a, -d, or -g modifiers. Multiple modifiers are allowed; if no modifiers are specified, all coverage types are indicated.

Example: -multicount=f-gt

This turns the fsm and toggle coverage types (single-bit by default) to multi-bit, and covergroup coverage type to single bit. All other coverage types remain unchanged.

Caution

The use of the vsim -coverstore automatically disables any "coverage save" commands that are executed as part of scripts or interactively. This ensures that storage space is not wasted on the writing of duplicate data into a traditional UCDB file. The "coverage attribute" commands can still be placed at the end of a simulation to modify data that is stored by the auto-save.

- -cppinstall <[gcc | g++] version>

  (optional) Specifies the version of the desired GNU compiler supported and distributed by Mentor Graphics.

  <[gcc | g++] version> — The version number of the GNU compiler to use. Use the same compiler version as specified on the sccom command line. For example:

  ```
  sccom -cppinstall 4.5.0
  ```

  Refer to “Supported Platforms and Compiler Versions” for a list of supported compilers.

- -cpppath <filename>

  (optional) Specifies the location of a g++ executable other than the default g++ compiler installed with Questa SIM. Overrides the CppPath variable in the *modelsim.ini* file.

- -debugdb=<db_pathname>

  (optional) Instructs Questa SIM to generate a database of connectivity information to be used for post-sim debug in the Dataflow and Schematic windows. The database pathname should have a *.dbg* extension. If a database pathname is not specified, Questa SIM creates a database file named *vsim.dbg* in the current directory.

  An existing *.dbg* file will be reused and a note printed to the transcript when the -debugdb switch is specified and your design has not changed since the database was created.

  Refer to “Post-Simulation Debug Flow Details” for more information.

- -default_radix <radix>

  (optional) Sets the default radix for the simulation and overrides the DefaultRadix preference variable. <radix> may be any of the following: ascii, binary, decimal, hexadecimal, octal, symbolic, unsigned.

- -defaultstdlogicinittoz

  (optional) Sets the default VHDL initialization of std_logic to "Z" (high impedance) for ports of type OUT and INOUT. IEEE Std 1076-1987 VHDL Language Reference Manual

(LRM) compliant behavior is for std_logic to initialize to "U" (uninitialized) which is incompatible with the behavior expected by synthesis and hardware.

- -display <display_spec>

  (optional) Specifies the name of the display to use. Does not apply to Windows platforms.

  For example:

  ```
  -display :0
  ```

- -displaymsgmode both | tran | wlf

  (optional) Controls the transcription of $display system task messages to the transcript and/or the Message Viewer. Refer to the section "Message Viewer Window" in the User's Manual for more information and the displaymsgmode *.ini* file variable.

  both — outputs messages to both the transcript and the WLF file.

  tran — outputs messages only to the transcript, therefore they are not available in the Message Viewer. Default behavior.

  wlf — outputs messages only to the WLF file/Message Viewer, therefore they are not available in the transcript.

  The display system tasks displayed with this functionality include: $display, $strobe, $monitor, $write as well as the analogous file I/O tasks that write to STDOUT, such as $fwrite or $fdisplay.

- -do "<command_string>" | <do_file_name>

  (optional) Instructs vsim to use the command(s) specified by <command_string> or the DO file named by <do_file_name> rather than the startup file specified in the *.ini* file, if any. Multiple commands can be specified as a semi-colon (;) separated list. You can also specify multiple instances of -do "<command_string>" on the same command line. The commands are joined together in the order specified.

  For example:

  ```
  vsim -do "force clk 0 0, 1 10 -r 20" top -wlf top.wlf /
  -do "testfile.do" -do "run -all"
  ```

  will turn into the following script:

  ```
  "force clk 0 0, 1 10 -r 20; do testfile.do; run -all"
  ```

- -donotcollapsepartiallydriven

  (optional) Prevents the collapse of partially driven and undriven output ports during optimization. Prevents incorrect values that can occur when collapsed.

- +dumpports+collapse | +dumpports+nocollapse

  (optional) Determines whether vectors (VCD id entries) in dumpports output are collapsed or not. The default behavior is collapsed, and can be changed by setting the DumpportsCollapse variable in the *modelsim.ini* file.

- +dumpports+direction

  (optional) Modifies the format of extended VCD files to contain direction information.

- +dumpports+no_strength_range

  (optional) Ignores strength ranges when resolving driver values for an extended VCD file. This argument is an extension to the IEEE 1364 specification. Refer to "Resolving Values" for additional information.

- +dumpports+unique

  (optional) Generates unique VCD variable names for ports in a VCD file even if those ports are connected to the same collapsed net.

- -elab <filename>

  (optional) Creates an elaboration file for use with -load_elab. Refer to "Simulating with an Elaboration File" for more information.

- -elab_cont <filename>

  (optional) Creates an elaboration file for use with -load_elab and then continues the simulation.

- -elab_defer_fli

  (optional) Defers the initialization of FLI models until the load of the elaboration file. Use this argument along with -elab to create elaboration files for designs with FLI models that don't support checkpoint/restore. Note that FLI models sensitive to design load ordering may still not work correctly even if you use this argument.

- -enumfirstinit

  (optional) Initializes enum variables in SystemVerilog using the leftmost value as the default. You must also use the argument with the vlog command in order to implement this initialization behavior. Specify the EnumBaseInit variable as 0 in the *modelsim.ini* file to set this as a permanent default.

- -error <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "error." Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -extendedtogglemode 1|2|3

  (optional) Changes the level of support for extended toggles for the simulation. The levels of support are:

  1 — 0L->1H & 1H->0L & any one 'Z' transition (to/from 'Z')

  2 — 0L->1H & 1H->0L & one transition to 'Z' & one transition from 'Z'

  3 — 0L->1H & 1H->0L & all 'Z' transitions

  Edit the ExtendedToggleMode variable in the *modelsim.ini* file to set a permanent default.

- -f <filename>

  (optional) Specifies a file with more vsim command arguments. Allows complex argument strings to be reused without retyping.

  Refer to the section "Argument Files" for more information.

- -fatal <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "fatal." Edit the fatal variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -feccountlimit [<n> | 0]

  (optional) Limits the number of counts that are tracked for Focused Expression Coverage. Specifically, when a bin has reached the specified count, coverage will ignore further tracking of the inputs linked to the bin.

Note

If you change this value from the default you may affect simulation performance.

<n> — Specifies the count limit for FEC. The default is 1

0 — Specifies an unlimited count

- -filemap_elab <HDLfilename>=<NEWfilename>

  (optional) Defines a file mapping during -load_elab that lets you change the stimulus. Refer to "Simulating with an Elaboration File" for more information.

- -fsmdebug

  (optional) Enables visualization of FSMs in the GUI. You must specify this argument to view FSM information the GUI.

  Refer to the section "Viewing FSM Information in the GUI" for more information.

- -g <Name>=<Value> …

  (optional) Assigns a value to all specified VHDL generics and Verilog parameters that have not received explicit values in generic maps, instantiations, or via defparams (such as top-level generics/parameters and generics/parameters that would otherwise receive their default values).

  Multiple -g options are allowed, one for each generic/parameter, specified as a space separated list.

  <Name> — Name of a generic/parameter, exactly as it appears in the VHDL source (case is ignored) or Verilog source. Name may be prefixed with a relative or absolute hierarchical path to select generics in an instance-specific manner. For example, specifying -g/top/u1/tpd=20ns on the command line would affect only the *tpd* generic on the */top/u1* instance, assigning it a value of 20ns. Specifying -gu1/tpd=20ns affects the *tpd* generic on all instances named *u1*. Specifying-gtpd=20ns affects all generics named *tpd*.

<Value> — Specifies an appropriate value for the declared data type of a VHDL generic or any legal value for a Verilog parameter. Make sure the value you specify for a VHDL generic is appropriate for VHDL declared data types. Integers are treated as signed values. For example, -gp=-10 overwrites the parameter p with the signed value of -10.

If more than one -g option selects a given generic the most explicit specification takes precedence. For example,

```
vsim -g/top/ram/u1/tpd_hl=10ns -gtpd_hl=15ns top
```

This command sets *tpd_hl* to 10ns for the */top/ram/u1* instance. However, all other *tpd_hl* generics on other instances will be set to 15ns.

Limitation: In general, generics/parameters of composite type (arrays and records) cannot be set from the command line. However, you can set string arrays, std_logic vectors, and bit vectors if they can be set using a quoted string. For example,

```
-gstrgen="This is a string"
-gslv="01001110"
```

The quotation marks (" ") must make it into vsim as part of the string because the type of the value must be determinable outside of any context. Therefore, when entering this command from a shell, put single quotes (' ') around the string. For example:

```
-gstrgen='"This is a string"'
```

If working within the Questa SIM GUI, you would enter the command as follows:

```
{-gstrgen="This is a string"}
```

You can also enclose the value escaped quotes (\"), for example:

```
-gstrgen=\"This is a string\"
```

- -G<Name>=<Value> …

  (optional) Same as -g (see above) except that it will also override generics/parameters that received explicit values in generic maps, instantiations, or from defparams.

  This argument is the only way for you to alter the generic/parameter, such as its length, (other than its value) after the design has been loaded.

  <Name> — Name of a generic/parameter, exactly as it appears in the VHDL source (case is ignored) or Verilog source. Name may be prefixed with a relative or absolute hierarchical path to select generics in an instance-specific manner. For example, specifying -G/top/u1/tpd=20ns on the command line would affect only the *tpd* generic on the */top/u1* instance, assigning it a value of 20ns. Specifying -Gu1/tpd=20ns affects the *tpd* generic on all instances named *u1*. Specifying -Gtpd=20ns affects all generics named *tpd*.

  <Value> — Specifies an appropriate value for the declared data type of a VHDL generic or any legal value for a Verilog parameter. Make sure the value you specify for a VHDL generic is appropriate for VHDL declared data types. Integers are treated

as signed values. For example, -Gp=-10 overwrites the parameter p with the signed value of -10.

- -gblso <shared_obj>[,<shared_obj>]

  (optional) Open the specified shared object(s) with global symbol visibility. Essentially all data and functions are exported from the specified shared object and are available to be referenced and used by other shared objects. If you specify multiple, comma-separated, shared objects, they will be merged internally and then loaded as a single shared object. You can also specify this argument with the GlobalSharedObjectsList variable in the *modelsim.ini* file.

- -geometry <geometry_spec>

  (optional) Specifies the size and location of the main window. Where <geometry_spec> is of the form:

  ```
  WxH+X+Y
  ```

- -gui

  (optional) Starts the Questa SIM GUI without loading a design and redirects the standard output (stdout) to the GUI Transcript window.

- -help

  (optional) Sends the arguments and syntax for vsim to the transcript.

- -i

  (optional) Specifies that the simulator be run in interactive mode.

- -immedassert | -noimmedassert

  (optional) Controls the simulation of immediate assertions (Verilog and VHDL). By default immediate assertions are simulated. Use the -noimmedassert option to disable simulation of immediate assertions. Similarly, use -immedassert to enable simulation of immediate assertions called. If you disable simulation of immediate assertions, they do not show up in assertion browser or reports.

  You can also specify this argument with the SimulateImmedAsserts variable in the modelsim.ini file.

- +initregNBA | +noinitregNBA

  (optional) Controls whether +initreg settings applied to registers of sequential UDPs should be non-blocking. This is useful when continuous assignments overwrite register initialization.

  +initregNBA — (default) enables this functionality.

  +noinitregNBA — disables this functionality.

- -installcolormap

  (optional) For UNIX only. Causes vsim to use its own colormap so as not to hog all the colors on the display. This is similar to the -install switch on Netscape.

- -keeploaded

  (optional) Prevents the simulator from unloading/reloading any HDL interface shared libraries when it restarts or loads a new design. The shared libraries will remain loaded at their current positions. User application code in the shared libraries must reset its internal state during a restart in order for this to work effectively.

- -keeploadedrestart

  (optional) Prevents the simulator from unloading/reloading any HDL interface shared libraries during a restart. The shared libraries will remain loaded at their current positions. User application code in the shared libraries must reset its internal state during a restart in order for this to work effectively.

  We recommend using this option if you'll be doing warm restores after a restart and the user application code has set callbacks in the simulator. Otherwise, the callback function pointers might not be valid if the shared library is loaded into a new position.

- -keepstdout

  (optional) For use with foreign programs. Instructs the simulator to not redirect the stdout stream to the Main window.

- -logfile <filename> | -l <filename> | -nolog

  (optional) Controls saving of transcript data during batch and regular simulations.

  -logfile <filename> — Saves transcript data to <filename>. Can be abbreviated to -l <filename>. Overrides the default transcript file creation set with the the TranscriptFile or BatchTranscriptFile *modelsim.ini* variables. You can also specify "stdout" or "stderr" for <filename>.

  -nolog — Disables transcript file creation. Overrides the TranscriptFile or BatchTranscriptFile variables set in the *modelsim.ini* file.

  Refer to "Batch Mode" for more information about saving transcript data.

- -L <library_name> …

  (optional) Specifies the library to search for design units instantiated from Verilog and for VHDL default component binding. Refer to "Library Usage" for more information. If multiple libraries are specified, each must be preceded by the -L option. Libraries are searched in the order in which they appear on the command line.

- -ldflags <"linkopts">

  (optional) Specify in quotes any options for linking auto compiled DPI object files. Multiple occurrences are supported.

- -learn <root_file_name>

  (optional) Specifies that you want the simulator to generate control files for retaining the proper level of visibility when performing an optimized simulation. The files generated are:

```
top_pli_learn.acc
top_pli_learn.ocf
top_pli_learn.ocm
```

Refer to "Preserving Design Visibility with the Learn Flow" in the User's Manual for more information.

It is suggested that you specify this option with -voptargs=+acc to enable as much visibility as possible to allow for functional correctness of your design.

- -Lf <library_name> …

  (optional) Same as -L but libraries are searched before 'uselib directives. Refer to "Library Usage" for more information.

- -L mtiPA

  (optional) Use library of precompiled Power Aware behavioral models.

- -lib <libname>

  (optional) Specifies the default working library where vsim will look for the design unit(s). Default is "work".

- <license_option>

  (optional) Restricts the search of the license manager. Use one of the license options listed below.

  You can specify a license option only when invoking vsim from a UNIX/Linux shell command line, DOS command shell command line, or a Target for a Windows desktop shortcut. If you specify a license option from the GUI (such as the GUI prompt, on Restart, or warm Restore) it will not have the intended effect.

<license_option>	**Description**
-lic_lnl_only	check out msimhdlsim license only
-lic_mixed_only	check out msimhdlsim/msimhdlmix licenses only
-lic_no_lnl	exclude msimhdlsim license
-lic_no_mix	exclude msimhdlmix license
-lic_no_slvhdl	exclude qhsimvh license
-lic_no_slvlog	exclude qhsimvl license
-lic_noqueue	do not wait in queue when license is unavailable
-lic_plus[1]	check out PLUS (VHDL and Verilog) license immediately after invocation
-lic_vhdl[1]	check out VHDL license immediately after invocation
-lic_vlog[1]	check out VLOG license immediately after invocation

1. This license feature is reserved upon vsim invocation.

You can also specify these options with the License variable in the *modelsim.ini* file. As with the command line switches, these settings will only affect the initial invocation of vsim and will not affect Restart or Restore.

Note

Note that settings made from the command line are additive to options set in the License variable.

For a complete list of license features and descriptions, refer to "License Feature Names" in the *Installation and Licensing Guide*.

- -load_elab <filename>

  (optional) Loads an elaboration file that was created with -elab. Refer to "Simulating with an Elaboration File" for more information.

  You can not use this switch with any form of the -memprof switch. To analyze memory usage when simulating an elaboration file, you will need to use the profile on command. For example:

  ```
  vsim -load_elab top.elab -do
     "profile on -m -fileonly top_mem_run.rpt; run -all"
  ```

- -memprof

  (optional) Causes memory allocation data to be collected during elaboration and simulation. Shows what part of the design is using memory.

- -memprof+call

  (optional) Unwinds the call stack and collects the call tree information. At the VSIM prompt, call stack collection can also be turned on with profile option collect_calltrees on and off with profile option collect_calltrees off.

- -memprof+file=<filename>

  (optional) Saves memory profile data to the named file and makes the data available for viewing and reporting during the current simulation. The file can be used for archival or comparison purposes.

- -memprof+fileonly=<filename>

  (optional) Saves memory profile data to the named file only. The file can be read in later with the profile reload command for analysis. This mode is useful for large designs, when the design plus internal profiling data would use up too much memory.

- -mlopt

  This option improves the optimization of mixed-language nets that cross Verilog-VHDL language boundaries. This primarily impacts areas where a large number of highly active nets, such as clocks, cross Verilog-VHDL language boundaries multiple times.

In some scenarios, there may be differences in simulation results. The differences primarily relate to initial values on mixed-language nets. The most common difference will be VHDL nets that may have an initial value of 'X' instead of 'U'.

Limitations:

- This option is ignored when doing power aware simulation.
- This option will be ignored when doing multi-core (mc2) simulation.
- SDF annotation of interconnect and port delays onto mixed-language nets is not supported. Attempting to annotate delays when -mlopt is specified results in errors.
- This option is only supported for VHDL std_logic and std_ulogic based types (including arrays where the elements are of these types), and Verilog logic types.
- VHDL type conversions and conversion functions are not allowed on a mixed language net when -mlopt is specified.
- This option is ignored if the design has analog components.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable. Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems the path separator should be a forward slash (/).

- -msgfile <filename>

  (optional) Designates an alternative file for recording error messages. An alternate file may also be specified by the ErrorFile modelsim.ini variable. By default, error messages are output to the file specified by the TranscriptFile variable in the *modelsim.ini* file (refer to "Creating a Transcript File").

- -msglimit [all, | none,] <msgNumber>[, <msgNumber>,…]

  (optional) Limits the number of iterations of the specified message(s) to five (the default count) then suppresses all new instances. Refer to "Suppression of Warning Messages" for more information.

  all — Limits all messages to the default count except the specified msgNumber(s).

  none — Limits only the specified msgNumber(s) to the default count.

  <msgNumber>[,<msgNumber>,…] — Specifies the message number(s) to limit to five iterations. Multiple messages are specified as a comma-separated list.

  For example:

  vsim -msglimit all, *<msgID1>*, *<msgID2>*

  Limits all messages to the default count except *msgID1* and *msgID2*.

  vsim -msglimit none, *<msgID1>*, *<msgID2>*

  Limits only *msgID1* and *msgID2* to the default count.

- -msglimitcount <limit_value> -msglimit [all, | none,] <msgNumber>[, <msgNumber>...]

  (optional) Limits the reporting of listed messages to user defined limit_value. Overrides the MsgLimitCount variable in the *modelsim.ini* file.

- -msgmode both | tran | wlf

  (optional) Specifies the location(s) for the simulator to output elaboration and runtime messages.

  both — outputs messages to both the transcript and the WLF file.

  tran — (default) outputs messages only to the transcript, therefore they are not available in the Message Viewer or the Results Analysis windows.

  wlf — outputs messages only to the WLF file/Message Viewer and Results Analysis windows, therefore they are not available in the transcript.

  Refer to the section "Message Viewer Window" in the User's Manual for more information.

- -multisource_delay min | max | latest

  (optional) Controls the handling of multiple PORT or INTERCONNECT constructs that terminate at the same port. By default, the Module Input Port Delay (MIPD) is set to the max value encountered in the SDF file. Alternatively, you may choose the min or latest of the values. If you have a Verilog design and want to model multiple interconnect paths independently, use the +multisource_int_delays argument.

- +multisource_int_delays

  (optional) Enables multisource interconnect delay with pulse handling and transport delay behavior. Works for both Verilog and VITAL cells.

  Use this argument when you have interconnect data in your SDF file and you want the delay on each interconnect path modeled independently. Pulse handling is configured using the +pulse_int_e and +pulse_int_r switches (described below).

  The +multisource_int_delays argument cannot be used if you compiled using the -novital argument to vcom. The -novital argument instructs vcom to implement VITAL functionality using VHDL code instead of accelerated code, and multisource interconnect delays cannot be implemented purely within VHDL.

- -mvchome <path>

  (required for use of Questa Verification IPs, unless path is specified in *modelsim.ini* file) Specifies the location where the Questa Verification IPs are installed. Overrides any path set with the MvcHome *modelsim.ini* variable.

- -name <name>

  (optional) Specifies the application name used by the interpreter for send commands. This does not affect the title of the window.

- -noassertcover

  (optional) Disables extended count information for assertions. This option may also be specified with the AssertionCover variable in the *modelsim.ini* file.

- -noassertdebug

  (optional) Disables assertion pass reporting and debug options such as assertion thread viewing (ATV), HDL failed expression analysis, extended count information, and causality traceback. This option may also be specified with the AssertionDebug variable in the *modelsim.ini* file.

- -noassume

  (optional) Disables simulation of PSL and SystemVerilog assume directives. This option may also be specified with the SimulateAssumeDirectives variable in the *modelsim.ini* file. Refer to "Processing Assume Directives" for more information.

- -no_autoacc

  (optional) Prevents vsim from automatically passing the +acc switch to vopt. By specifying this argument you can prevent vopt from opening any Verilog PLI modules for accessibility. You can pass specific +acc options to vopt by using the -voptargs argument.

- -noautoldlibpath

  (optional) Disables the default internal setting of LD_LIBRARY_PATH, enabling you to set it yourself. Use this argument to make sure that LD_LIBRARY_PATH is not set automatically while you are using the GUI,

- -nocapacity

  (optional) Disables the display of both coarse-grain and fine-grain analysis of memory capacity.

- -nocompress

  (optional) Causes VSIM to create uncompressed checkpoint files. This option may also be specified with the CheckpointCompressMode variable in the *modelsim.ini* file.

- -nocrossautobins[=[cond|uncond]]

  (optional) Prevents the automatic generation of cross bins for covergroups. Cross bins are excluded from coverage computation and coverage reports.

  cond — When "cond" is specified, the auto cross bins are not generated only if there are one or more coverable user defined cross bins.

  uncond — When "uncond" is specified, the auto cross bins generation will be always skipped.

- -noimmedca

  (optional) Causes Verilog event ordering to occur without enforced prioritization—continuous assignments and primitives are not run before other normal priority processes scheduled in the same iteration. Use this argument to prevent the default event ordering where continuous assignments and primitives are run with "immediate priority." You may also set even ordering with the ImmediateContinuousAssign variable in the *modelsim.ini* file.

- +no_notifier

  (optional) Disables the toggling of the notifier register argument of all timing check system tasks. By default, the notifier is toggled when there is a timing check violation, and the notifier usually causes a UDP to propagate an X. This argument suppresses X propagation in both Verilog and VITAL for the entire design.

  You can suppress X propagation on individual instances using the tcheck_set command.

- -nopsl

  (optional) Instructs Questa SIM to ignore any PSL temporal directives (assertions, covers, endpoints) that were compiled with the design. Any HDL code present in the PSL code will remain active. (By default vsim automatically invokes the PSL assertion engine at runtime if any assertions were compiled with the design.)

- +nospecify

  (optional) Disables specify path delays and timing checks in Verilog.

- -nostdout

  (optional) Directs all output to the transcript only when in command line and batch mode. Prevents duplication of I/O between the shell and the transcript file. Has no affect on interactive GUI mode. Refer to “Batch Mode” for information about batch mode usage.

- -nosva

  (optional) Instructs Questa SIM to ignore any SystemVerilog concurrent assertions that were compiled with the design. By default, vsim automatically invokes the assertion engine at runtime if any assertions were compiled with the design.

- +no_tchk_msg

  (optional) Disables error messages generated when timing checks are violated. For Verilog, it disables messages issued by timing check system tasks. For VITAL, it overrides the MsgOn arguments and generics.

  Notifier registers are still toggled and may result in the propagation of Xs for timing check violations.

  You can disable individual messages using the tcheck_set command.

- -note <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "note." Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to “Message Severity Level” for more information.

- +notifier_ondetect

  (optional) Causes timing check notifier toggle generated X output values to be scheduled with zero delay.

- +notiftoggle01[+<seed>]

  (optional) Uses the metastable UDP evaluation of enabled Verilog cells in simulation. Default seed value is 0. See Approximating Metastability.

  NOTE: This argument is deprecated for future use. It is recommended to use -[no]randmetastable [+<seed>] instead.

- +notimingchecks | +ntcnotchks

  (optional) Disables Verilog timing checks. (This option sets the generic TimingChecksOn to FALSE for all VHDL Vital models with the Vital_level0 or Vital_level1 attribute. Generics with the name TimingChecksOn on non-VITAL models are unaffected.) By default, Verilog timing check system tasks ($setup, $hold,…) in specify blocks are enabled. For VITAL, the timing check default is controlled by the ASIC or FPGA vendor, but most default to enabled.

  Additionally, +ntcnotchks maintains the delay net delays necessitated by negative timing check limits. For this reason when using +ntcnotchks it is necessary to SDF annotate all timing check values.

  You can disable individual checks using the tcheck_set command.

- -notogglealias

  (optional) Prevents the storage of alias toggle nodes in the UCDB during the coverage save from simulation. Alias toggle nodes are stored by default with coverage save.

- -novopt

  (optional) Prevents Questa SIM from running the vopt command automatically. If you have the VoptFlow variable set to 1 (optimizations turned on) in the *modelsim.ini* file, vsim automatically runs vopt if you didn't invoke it manually. If you specify this argument, you should be sure to specify it with your compilation command (vcom or vlog). Use of this switch is not recommended except when using Power Aware simulation together with QuestaADMS. If this switch is used, the following warning is produced:

  ```
  ** Warning: (vsim-8891) All optimizations are turned off because the
  -novopt switch is in effect. This will cause your simulation to run
  very slowly. If you are using this switch to preserve visibility for
  Debug or PLI features, please see the User's Manual section on
  Preserving Object Visibility with vopt.
  ```

- -nowiremodelforce

  (optional) Restores the force command to previous usage (prior to version 10.0b) where an input port cannot be forced directly if it is mapped at a higher level in VHDL and mixed models. Signals must be forced at the top of the hierarchy connected to the input port.

- -optionset <optionset_name>

  (optional) Calls an optionset as defined in the *modelsim.ini file*. Refer to the section "Optionsets" in the Reference Manual for more information.

- -OVMdebug

  (optional) Enables the creation of information for using the OVM-aware debugging windows in the GUI. Refer to the section "OVM-Aware Debug" in the User's Manual.

- -ovmtestname

  (optional) Used only with the higher performance, Coverstore merge flow. Specifies that the +OVM_TESTNAME name is used both for the testnames and for the names of the data files produced. SV seed values (specified with -sv_seed) are also added in coverstore testnames and data file names, by default.

- +OVM_TESTNAME=<testname>

  (optional) Specifies the testname to be used for OVM designs.

- -pa

  (optional) Performs the simulation in Power Aware mode. Specific to Power Aware simulation.

- -pa_allowtimezeroevent[=all]

  (optional) Enables corruption, isolation, or release of signals at time 0 for the following events: pa_corrupt_register, pa_iso_on and ap_iso_off. This base functionality is on by default, you can disable it with the -pa_disabletimezeroevent switch.

  =all — enables corruption, isolation, or release of signals at time 0 for all Named Events in Power Aware.

- -pa_debugdir <directory>

  Specifies the location for the writing and retrieval of post-simulation debug information for power aware simulations. Refer to the section "Power Aware Simulation Debug" in the *Power Aware Simulation User's Manual* for more information.

- -pa_disabletimezeroevent

  (optional) Disables the default behavior of the -pa_allowtimezeroevent switch.

- -pa_gls <testbench_top>

  (optional) Enables gate-level simulation for Power Aware, performed on the top-of-design test bench (testbench_top).

- -pa_highlight

  (optional) Enables visual indication (highlighting) of power states of signals viewed in the Wave window. Waveforms are highlighted during the interval when they are corrupted, isolated, or biased.

- -pa_lib <pathname>

  (optional) Specifies the library location where Power Aware information from the vopt command is stored.

- -pa_togglelimit=<n>

  (optional) Instructs vsim to discontinue reporting an error (8906) for each signal that toggles during power off and issue note (8922) instead. The default value is 5. This switch only impacts your simulation if you used -pa_checks=t or -pa_checks=it with your vopt command.

- -pa_top <pathname>

  (optional) Changes the hierarchical prefix to the DUT of a Power Aware analysis at simulation, where <pathname> is the full path to the Verilog or VHDL design element under test (including the instantiation name). Note that you use this argument with the vopt -pa_defertop command and argument for the first time on the specified DUT. You can then use this argument to change Questa SIM Power Aware simulation at runtime without having to re-run vopt on the same DUT. That is, you can run a simulation with a different top hierarchy to the DUT that was passed with vopt (by changing the root of DUT hierarchy to <pathname>) to create Power Aware results without running another vopt command.

- -pduignore[=<instpath>]

  Ignore Preoptimized Design Unit (black-box). If <instpath> is not specified all PDUs found in compiled libraries will be ignored. Otherwise the PDU specified by <instpath> will be ignored. This option may be specified multiple times with different <instpath>s. Equivalent to the deprecated “-ignore_bbox” option.

- -permissive

  (optional) Allows messages in the LRM group of error messages to be downgraded to a warning.

  You can produce a complete list by executing the command:

  ```
  verror -kind vsim -permissive
  ```

- -postsimdataflow

  (optional) Makes Dataflow window available for post simulation debug operations. By default, the Dataflow window is not available for post-sim debug.

- -printsimstats[=[<val>][v]]

  (optional) Prints the output of the simstats command to the transcript at the end of simulation before exiting. <val> is 0 - disables simstats, 1(default) - prints stats at the end of simulation, 2 - prints out stats at the end of each run command and simulation. v- prints out verbose statistics, including the checkout time.

  Each performance statistic is printed with its related units on a separate line. Edit the PrintSimStats variable in the *modelsim.ini* file to set the simulation to print the simstats data by default.

  The command vsim -printsimstats=v is equivalent to vsim -stats=perf+verbose. The command vsim -printsimstats=2v is equivalent to vsim -stats=perf+verbose+eor.

- -psl

  (optional) Instructs Questa SIM to elaborate all PSL assertion directives. Overrides the SimulatePSL *modelsim.ini* file setting. See also vsim -nopsl.

- -psloneattempt

  (optional) Force a single attempt to start at the beginning of simulation in order to test PSL directives with top level "always/never" properties. As per strict 1850-2005 PSL LRM, an always/never property can either pass or fail. However, by default, QuestaSim reports multiple passes and/or failures, which corresponds to multiple attempts made while executing a top level "always/never" property. With this switch, the directive will either match (pass), fail, or vacuously-match (provided it is not disabled/aborted). If the "always/never" property fails, the directive is immediately considered a failure and the simulation will not go further. If there is no failure (or disable/abort) until end of simulation then a match (pass) is reported. You can turn this feature on permanently using the PslOneAttempt variable in the *modelsim.ini* file.

- -pslinfinitethreshold=<integer>

  (optional) Specifies the number of clock ticks that represent infinite clock ticks. Only affects PSL strong operators, namely: eventually!, until! and until_!. If the end of simulation is reached and an active strong-property has not clocked this number of clock ticks, neither pass nor fail (i.e. vacuous match) is returned; else, respective fail/pass is returned. The default value is '0' (zero) which effectively does not check for clock tick condition. You can turn on this feature using this switch or the PslInfinityThreshold in the *modelsim.ini* file.

- +pulse_int_e/<percent>

  (optional) Controls how pulses are propagated through interconnect delays, where <percent> is a number between 0 and 100 that specifies the error limit as a percentage of the interconnect delay. Used in conjunction with +multisource_int_delays (see above). This option works for both Verilog and VITAL cells, though the destination of the interconnect must be a Verilog cell. The source may be VITAL or Verilog.

  A pulse greater than or equal to the error limit propagates to the output in transport mode (transport mode allows multiple pending transitions on an output). A pulse less than the error limit and greater than or equal to the rejection limit (see +pulse_int_r/<percent> below) propagates to the output as an X. If the rejection limit is not specified, then it defaults to the error limit. For example, consider an interconnect delay of 10 along with a +pulse_int_e/80 option. The error limit is 80% of 10 and the rejection limit defaults to 80% of 10. This results in the propagation of pulses greater than or equal to 8, while all other pulses are filtered.

- +pulse_int_r/<percent>

  (optional) Controls how pulses are propagated through interconnect delays, where <percent> is a number between 0 and 100 that specifies the rejection limit as a percentage of the interconnect delay. This option works for both Verilog and VITAL cells, though the destination of the interconnect must be a Verilog cell. The source may be VITAL or Verilog.

A pulse less than the rejection limit is filtered. If the error limit is not specified by +pulse_int_e then it defaults to the rejection limit.

- -quiet

  (optional) Disable 'Loading' messages during batch-mode simulation.

- -[no]randmetastable [+<seed>]

  (optional) Enable or disable random insertion of 0 or 1 values (1'b0 or 1'b1) on the output of a Verilog library cell when a timing violation is encountered. See Approximating Metastability.

  +<seed> — First value inserted on cell output to begin randomization (0 or 1). Default value is 0.

- -restore <filename>

  (optional) Specifies that vsim is to restore a simulation saved with the checkpoint command.

  You must restore vsim under the same environment in which you did the checkpoint. This means not only the same type of machine and OS and at least the same memory size, but also the same vsim environment such as GUI vs. command line mode.

- -runinit

  (optional) Initializes non-trivial static SystemVerilog variables, for example expressions involving other variables and function calls, before displaying the simulation prompt.

- +sdf_iopath_to_prim_ok

  (optional) Prevents vsim from issuing an error when it cannot locate specify path delays to annotate. If you specify this argument, IOPATH statements are annotated to the primitive driving the destination port if a corresponding specify path is not found. Refer to "SDF to Verilog Construct Matching" for additional information.

- -sdfmin | -sdftyp | -sdfmax[@<delayScale>] [<instance>=]<sdf_filename>

  (optional) Annotates VITAL or Verilog cells in the specified SDF file (a Standard Delay Format file) with minimum, typical, or maximum timing. Can also specify instances under VHDL generates as the SDF back-annotation point.

  @<delayScale> — scales all values by the specified value. For example, if you specify -sdfmax@1.5, all maximum values in the SDF file are scaled to 150% of their original value.

  Do not use this option if you scaled the SDF file while using the sdfcom command.

  <instance>= — specifies a specific instance for the associated SDF file. Use this when not performing backannotation at the top level.

  <sdf_filename> — specifies the file containing the SDF information.

- -sdfminr | -sdftypr | -sdfmaxr[@<delayScale>] [<instance>=]<sdf_filename>

  (optional) Specifies when an instance of a Preoptimized Design Unit (vopt -pdu, formerly -bbox) with an associated default SDF file is to be re-annotated with minimum, typical, or maximum timing from the specified SDF file.

@<delayScale> — scales all values by the specified value. For example, if you specify -sdfmax@1.5, all maximum values in the SDF file are scaled to 150% of their original value.

Do not use this option if you scaled the SDF file while using the sdfcom command.

<instance>= — specifies a specific instance for the associated SDF file. Use this when not performing back-annotation at the top level.

<sdf_filename> — specifies the file containing the SDF information.

Note

The simulator assumes that the instance/timing object hierarchy in the new SDF file is compatible with the SDF file specified together with vopt -pdu (black-boxing).

The following is a simple usage flow:

Assume module top contains three instances (u1, u2, and u3) of a Preoptimized Design Unit named pduMod.

```
vlib work
vlog pduMod.v
```

Preoptimize pduMod and annotate with *sdf1*.

```
vopt -pdu pduMod -o pduMod_opt -sdfmin pduMod=sdf1
vlog top.v
```

Use the default SDF file *sdf1* for the PDU instance of u1, but override the SDF for u2 and u3.

```
vsim top +sdf_verbose -sdftypr /top/u2=sdf2 -sdfmaxr /top/u3=sdf3
run -all
```

- -sdfmaxerrors <n>

  (optional) Controls the number of Verilog SDF missing instance messages to be generated before terminating vsim. <n> is the maximum number of missing instance error messages to be emitted. The default number is 5.

- -sdfnoerror

  (optional) Errors issued by the SDF annotator while loading the design prevent the simulation from continuing, whereas warnings do not. Changes SDF errors to warnings so that the simulation can continue.

- -sdfnowarn

  (optional) Disables warnings from the SDF reader. Refer to "VHDL Simulation" for an additional discussion of SDF.

- -sdfreport=<filename>

  (optional) Produces a report at the location of <filename> containing information about unannotated and partially-annotated specify path objects, specifically path delays and

timing checks. Refer to the section "Reporting Unannotated Specify Path Objects" in the User's Manual for more information.

- +sdf_report_unannotated_insts

  (optional) Enables error messages for any un-annotated Verilog instances with specify blocks or VHDL instances with VITAL timing generics that are under regions of SDF annotation.

- +sdf_verbose

  (optional) Turns on the verbose mode during SDF annotation. The Transcript window provides detailed warnings and summaries of the current annotation as well as information including the module name, source file name and line number.

- -stackcheck

  (optional) Enables runtime stack usage sanity checking. This argument causes vsim to add additional instrumentation at runtime to monitor the system stack usage. If the usage exceeds the reserved stack limit, vsim will report a fatal error. An uncaught stack overflow will lead to a potentially mysterious and random downstream crash.

- -stats [=[+ | -]<feature>[,[+ | -]<mode>]

  (optional) Controls display of statistics sent to a logfile, stdout, or the transcript. Specifying -stats without options sets the default features (cmd, msg, and time).

  When using the Two-Step Flow, the statistics results from the vopt subprocess are automatically integrated into the vsim stats report.

  Multiple features and modes for each instance of -stats are specified as a comma separated list. You can specify -stats multiple times on the command line, but only the last instance takes effect.

  You can specify -printsimstats and -stats on the same command line, however -stats will always override -printsimstats regardless of the order in which the options are specified.

  [+ | -] — Controls activation of the feature or mode where the plus character (+) enables the feature and the minus character (-) disables the feature. You can also enable a feature or mode by specifying a feature or mode without the plus (+) character. Setting this switch will add or subtract features and modes from the default settings "cmd,msg,time".

  Features

  all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none option. When specified in a string with other options, all is applied first.

  cmd — (default) Echo the command line.

  msg — (default) Display error and warning summary at the end of command execution.

  none — Disable all statistics features. Mutually exclusive with all option. When specified in a string with other options, none is applied first.

  perf — Display time and memory performance statistics.

time — (default) Display Start, End, and Elapsed times.

Modes

Modes can be set for a specific feature or globally for all features. To add or subtract a mode for a specific feature, specify using the plus (+) or minus (-) character with the feature, for example, vcover dump -stats=cmd+verbose,perf+list. To add or subtract a mode globally for all features, specify the modes in a comma-separated list, for example, vcover dump -stats=time,perf,list,-verbose. You cannot specify global and feature specific modes together.

eor — Print performance statistics at the end of each run command. Valid for use only with perf feature (-stats=perf+eor); it is invalid with other features.

kb — Print statistics in Kb units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

Note

By default, vsim prints the command line with the '-f filename' option. Prior to 10.3c, behavior was to print the command line with expanded arguments from '-f filename'. To enable the previous behavior, specify -stats=cmd+verbose.

Refer to Tool Statistics Messages for more information.

- -suppress <msg_number>[,<msg_number>,…]

  (optional) Prevents the specified message(s) from displaying. You cannot suppress Fatal or Internal messages. Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -sva

  (optional) Instructs Questa SIM to elaborate all SVA concurrent assertion directives. Overrides the **SimulateSVA** *modelsim.ini* file setting. See also vsim -nosva.

- -sync

  (optional) Executes all X server commands synchronously, so that errors are reported immediately. Does not apply to Windows platforms.

- -syncio | -nosyncio

  (optional) Controls buffering of text output to console and logfile. Both options are no-ops when vsim -batch is in effect.

  -syncio — (default) I/O is synchronous with simulation activity, always up-to-date.

  -nosyncio — Disables I/O synchronization. vsim will run faster by buffering (delaying) output.

- -t [<multiplier>]<time_unit>

  (optional) Specifies the simulator time resolution. <time_unit> must be one of the following:

```
fs, ps, ns, us, ms, sec
```

The default is 1ns; the optional <multiplier> may be 1, 10 or 100.

Note that there is no space between the multiplier and the unit (for example, 10fs, not 10 fs).

If you omit the -t argument, the default simulator time resolution depends on design type:

- In a VHDL design—the value specified for the Resolution variable in modelsim.ini is used.
- In a Verilog design with ‘timescale directives—the minimum specified time precision of all directives is used.
- In a Verilog design with no ‘timescale directives—the value specified for the Resolution variable in the modelsim.ini file is used.
- In a mixed design with VHDL on top—the value specified for the Resolution variable in the modelsim.ini file is used.
- In a mixed design with Verilog on top—
  - for Verilog modules not under a VHDL instance: the minimum value specified for their ‘timescale directives is used.
  - for Verilog modules under a VHDL instance: all their ‘timescale directives are ignored (the minimum value for ‘timescale directives in all modules not under a VHDL instance is used).

  If there are no ‘timescale directives in the design, the value specified for the Resolution variable in modelsim.ini is used.

  For SystemC designs, refer to “SystemC Time Unit and Simulator Resolution” for more information.

  For mixed-language designs with SystemC, refer to “Simulator Resolution Limit,” for more information.

Tip: After you have started a simulation, you can view the current simulator resolution by using the report command as follows:

report simulator state

- -tab <tabfile>

  (optional) Specifies the location of a Synopsys VCS “tab” file (.tab), which the simulator uses to automate the registration of PLI functions in the design.

  <tabfile> — The location of a *.tab* file contains information about PLI functions. The tool expects the *.tab* file to be based on Synopsys VCS version 7.2 syntax. Because the format for this file is non-standard, changes to the format are outside of the control of Mentor Graphics.

By specifying the location of a *.tab* file, you do not need to use the -no_autoacc switch, which prevents vopt from opening PLI modules for accessibility.

If you are using the Two-step optimization flow, the tool passes this information automatically to vopt, which uses the file to improve accessibility rules.

If you are using the Three-step optimization flow, you must specify this switch on both the vopt and vsim command lines.

- -tag <string>

  (optional) Specifies a string tag to append to foreign trace filenames. Used with the -trace_foreign <int> option. Used when running multiple traces in the same directory.

  Refer to the *Questa SIM FLI Reference* for more information.

- -tbxhvllint

  (optional. For use with Veloce emulator and TestBench-Xpress (TBX) verification accelerator) Enables TBX to identify delays that are encountered at runtime. Refer to the *TBX User's Guide* for more detailed usage information.

- -title <title>

  (optional) Specifies the title to appear for the Questa SIM Main window. If omitted the current Questa SIM version is the window title. Useful when running multiple simultaneous simulations. Text strings with spaces must be in quotes (for example, "my title").

- -togglecountlimit <int>

  (optional) Specifies the global toggle coverage count limit for toggle nodes in an entire simulation. Overrides the global value set by the ToggleCountLimit *modelsim.ini* variable. If used, it provides default limit values for any design units not compiled with either vlog -togglecountlimit or vcom –togglecountlimit. If any design units were compiled with those arguments, those values apply during simulation unless you use the toggle add –countlimit command to override the values. After the limit is reached, further activity on the node is ignored for toggle coverage. All possible transition edges must reach this count for the limit to take effect. For example, if you are collecting toggle data on 0->1 and 1->0 transitions, both transition counts must reach the limit. If you are collecting "full" data on 6 edge transitions, all 6 must reach the limit.

- -togglewidthlimit <int>

  (optional) Sets the maximum width of signals, <int>, that are automatically added to toggle coverage with the -cover t argument for vcom or vlog. Overrides the global value set by the ToggleWidthLimit *modelsim.ini* variable. If used, it provides default limit values for any design units not compiled with vlog –togglewidthlimit or vcom –togglewidthlimit.

- -trace_foreign <int>

  (optional) Creates two kinds of foreign interface traces: a log of what functions were called, with the value of the arguments, and the results returned; and a set of C-language files to replay what the foreign interface side did.

The purpose of the logfile is to aid the debugging of your PLI/VPI code. The primary purpose of the replay facility is to send the replay file to MTI support for debugging co-simulation problems, or debugging problems for which it is impractical to send the PLI/VPI code.

This also applies to FLI. See the *Questa SIM FLI Reference* for more information.

- -ucdbteststatusmsgfilter <TCL_subtext>

  (optional) Specifies a regular expression which — if matched when compared against all messages of severity level Failure/Fatal, or Error — prevents the status of that message from being propagated to the UCDB TESTSTATUS. <TCL_subtext> is a TCL style regular expression. If subtext contains spaces, quotes must surround the text. This can be set for multiple simulations using a *modelsim.ini* file variable. While valid for use with both VHDL and Verilog designs, this argument is perhaps most useful for keeping VHDL assertions which use “failure” to stop the simulation from being the worst severity in the simulation. For further information on the application of this argument, see the *Verification Management User’s Manual*.

- -udpcountlimit [<n> | 0]

  (optional) Limits the number of counts that are tracked for UDP Coverage. Specifically, when a bin has reached the specified count, coverage will ignore further tracking of the inputs linked to the bin.

Note

If you change this value from the default you may affect simulation performance.

<n> — Specifies the count limit for UDP coverage. The default is 1

0 — Specifies an unlimited count

- -undefsyms={<args>}

  (optional) Manages the undefined symbols in the shared libraries currently being loaded into the simulator. You can also manage undefined symbols with the UndefSyms *modelsim.ini* variable.

  {<args>}

  You must specify at least one argument.

  on — Enables automatic generation of stub definitions for undefined symbols and permits loading of the shared libraries despite the undefined symbols.

  off — (default) Disables loading of undefined symbols. Undefined symbols trigger an immediate shared library loading failure.

  verbose — Permits loading to the shared libraries despite the undefined symbols and reports the undefined symbols for each shared library.

- -usenonstdcoveragesavesysf

  (optional) Replaces implementation of the built-in, IEEE 1800 compliant system function with the non-standard variant, and thus affects all calls to $coverage_save(). The action of this switch is global.

- -uvmcontrol={<args>}

  (optional) Controls UVM-Aware debug features. These features work with either a standard Accelera-released open source toolkit or the pre-compiled UVM library package in Questa SIM.

  {<args>}

  You must specify at least one argument. You can enable or disable some arguments by prefixing the argument with a dash (-). Refer to the argument descriptions for more information.

  all — Enables all UVM-Aware functionality and debug options except disable and verbose. You must specify verbose separately.

  certe — Enables the integration of the elaborated design in the Certe tool. Disables Certe features when specified as -certe.

  disable — Prevents the UVM-Aware debug package from being loaded. Changes the results of randomized values in the simulator.

  msglog — Enables messages logged in UVM to be integrated into the Message Viewer. You must also enable wlf message logging by specifying tran or wlf with vsim -msgmode. Disables message logging when specified as -msglog

  none — Turns off all UVM-Aware debug features. Useful when multiple -uvmcontrol options are specified in a separate script, makefile or alias and you want to be sure all UVM debug features are turned off.

  struct — (default) Enables UVM component instances to appear in the Structure window. UVM instances appear under “uvm_root” in the Structure window. Disables Structure window support when specified as -struct.

  trlog — Enables or disables UVM transaction logging. Logs UVM transactions for viewing in the Wave window. Disables transaction logging when specified as -trlog.

  verbose — Sends UVM debug package information to the transcript. Does not affect functionality. Must be specified separately.

  Arguments may be specified as multiple instances of -uvmcontrol. Multiple arguments are specified as a comma separated list without spaces. For example,

  ```
  vsim -uvmcontrol=all,-trlog
  ```

  enables all UVM features except UVM transaction logging. Where arguments are in conflict, the last argument will override earlier arguments and a warning is issued.

  You can also control UVM-Aware debugging with the UVMControl modelsim.ini variable.

- -uvmtestname

  (optional) Used only with the higher performance, Coverstore merge flow. Specifies that the +UVM_TESTNAME name is used both for the testnames and for the names of the data files produced. SV seed values (specified with -sv_seed) are also added in coverstore testnames and data file names, by default.

- +UVM_TESTNAME=<testname>

  (optional) Defines the +UVM_TESTNAME test name to be used for UVM designs.

- -vcdstim [<instance>=]<filename>

  (optional) Specifies a VCD file from which to re-simulate the design. The VCD file must have been created in a previous Questa SIM simulation, which was excuted with the +dumpports+nocollapse option, then using the vcd dumpports command. Refer to "Using Extended VCD as Stimulus" for more information.

- -version

  (optional) Returns the version of the simulator as used by the licensing tools.

- -view [<alias_name>=]<WLF_filename>

  (optional) Specifies a wave log format (WLF) file for vsim to read. Allows you to use vsim to view the results from an open simulation (*vsim.wlf*) or an earlier saved simulation. The Structure, Objects, Wave, and List windows can be opened to look at the results stored in the WLF file (other Questa SIM windows will not show any information when you are viewing a dataset).

  <alias_name> — Specifies an alias for <WLF_file_name> where the default is to use the prefix of the WLF_filename. Wildcard characters are allowed.

  <WLF_file_name> — Specifies the pathname of a saved WLF file.

  See additional discussion in the Examples.

- -viewcov [<dataset_name>=]<UCDB_filename>

  (required for coverage view mode) Invokes vsim in the coverage view mode to display UCDB data.

- -visual <visual>

  (optional) Specifies the visual to use for the window. Does not apply to Windows platforms.

  Where <visual> may be:

  <class> <depth> — One of the following:

  {directcolor | grayscale | greyscale | pseudocolor | staticcolor | staticgray | staticgrey | truecolor}

  followed by:

  <depth> — Specifies how many bits per pixel are needed for the visual.

  default — Instructs the tool to use the default visual for the screen

<number> — Specifies a visual X identifier.

best <depth> — Instructs the tool to choose the best possible visual for the specified <depth>, where:

<depth> — Specifies how many bits per pixel are needed for the visual.

- +vlog_retain_on | +vlog_retain_off

  (optional) Enables or disables SDF RETAIN delay processing. +vlog_retain_on is the default behavior. Refer to the section "Retain Delay Behavior" in the User's Manual for more information.

- +vlog_retain_same2same_on | +vlog_retain_same2same_off

  (optional) Enables or disables SDF RETAIN delay processing of X insertion on outputs that do not change, but the causal inputs change. +vlog_retain_same2same_on is the default behavior. Refer to the section "Retain Delay Behavior" in the User's Manual for more information.

- -vopt

  (optional) Instructs vsim to run the vopt command automatically if vopt was not manually invoked. Not needed unless the VoptFlow variable has been set to 0 in the *modelsim.ini*. Refer to the chapter entitled "Optimizing Designs with vopt" for more information.

- -voptargs="<args>"

  (optional) Specifies arguments that vsim should pass to vopt when running vopt automatically. Any arguments must be separated by spaces, just as you would on the vopt command line.

  The primary purpose of this argument is to pass +acc arguments to vopt; the vopt -o <name> option is not permitted.

- -vopt_verbose

  (optional) Displays vopt messages in the Transcript window. By default, these messages are not displayed or saved when vopt is run using vsim.

- -warning <msg_number>[,<msg_number>,…]

  (optional) Changes the severity level of the specified message(s) to "warning." Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to "Message Severity Level" for more information.

- -warning error

  (optional) Reports all warnings as errors.

- -wlf <file_name>

  (optional) Specifies the name of the wave log format (WLF) file to create. The default file name is *vsim.wlf*. This option may also be specified with the WLFFilename variable in the *modelsim.ini* file.

- -wlfcachesize <n>

  (optional) Specifies the size in megabytes of the WLF reader cache. By default the cache size is set to zero. WLF reader caching caches blocks of the WLF file to reduce redundant file I/O. This should have significant benefit in slow network environments. This option may also be specified with the WLFCacheSize variable in the *modelsim.ini* file.

- -wlfcollapsedelta

  (default) Instructs Questa SIM to record values in the WLF file only at the end of each simulator delta step. Any sub-delta values are ignored. May dramatically reduce WLF file size. This option may also be specified with the WLFCollapseMode variable in the *modelsim.ini* file.

- -wlfcollapsetime

  (optional) Instructs Questa SIM to record values in the WLF file only at the end of each simulator time step. Any delta or sub-delta values are ignored. May dramatically reduce WLF file size. This option may also be specified with the WLFCollapseMode variable in the *modelsim.ini* file.

- -nowlfcollapse

  (optional) Instructs Questa SIM to preserve all events for each logged signal and their event order to the WLF file. May result in relatively larger WLF files. This option may also be specified with the WLFCollapseMode variable in the *modelsim.ini* file.

- -wlfcompress

  (default) Creates compressed WLF files. Use -nowlfcompress to turn off compression. This option may also be specified with the WLFCompress variable in the *modelsim.ini* file.

- -nowlfcompress

  (optional) Causes vsim to create uncompressed WLF files. WLF files are compressed by default in order to reduce file size. This may slow simulation speed by one to two percent. You may want to disable compression to speed up simulation or if you are experiencing problems with faulty data in the resulting WLF file. This option may also be specified with the WLFCompress variable in the *modelsim.ini* file.

- -wlfdeleteonquit

  (optional) Deletes the current simulation WLF file (*vsim.wlf*) automatically when the simulator exits. This option may also be specified with the WLFDeleteOnQuit variable in the *modelsim.ini* file.

- -nowlfdeleteonquit

  (default) Preserves the current simulation WLF file (vsim.wlf) when the simulator exits. This option may also be specified with the WLFDeleteOnQuit variable in the *modelsim.ini* file.

- -wlflock

  (optional) Locks a WLF file. An invocation of Questa SIM will not overwrite a WLF file that is being written by a different invocation.

- -nowlflock

  (optional) Disables WLF file locking. This will prevent vsim from checking whether a WLF file is locked prior to opening it as well as preventing vsim from attempting to lock a WLF once it has been opened.

- -wlfopt

  (default, optional) Optimizes the WLF file. Enables faster display of waveforms in the Wave window when the display is zoomed out to display a larger time range. This option may also be specified with the WLFOptimize variable in the *modelsim.ini* file. Refer to the section "Multitreading on Linux Platforms" in the User's Manual for more information.

- -nowlfopt

  (optional) Disables optimization of waveform display in the Wave window. This option may also be specified with the WLFOptimize variable in the *modelsim.ini* file.

- -wlfsimcachesize <n>

  (optional) Specifies the size in megabytes of the WLF reader cache for the current simulation dataset only. By default the cache size is set to zero. This makes it easier to set different sizes for the WLF reader cache used during simulation and those used during post-simulation debug. WLF reader caching caches blocks of the WLF file to reduce redundant file I/O. If neither the -wlfsimcachesize switch nor the WLFSimCacheSize *modelsim.ini* variable are specified, the -wlfcachesize switch or the WLFCacheSize *modelsim.ini* variable settings will be used.

- -wlfslim <size>

  (optional) Specifies a size restriction for the event portion of the WLF file.

  size — an integer, in megabytes, where the default is 0, which implies an unlimited size.

Note

Note that a WLF file contains event, header, and symbol portions. The size restriction is placed on the event portion only. Consequently, the resulting file will be larger than the specified size.

If used in conjunction with -wlftlim, the more restrictive of the limits takes precedence.

This option may also be specified with the WLFSizeLimit variable in the *modelsim.ini* file. (See Limiting the WLF File Size.)

- -wlftlim <duration>

  (optional) Specifies the duration of simulation time for WLF file recording. The default is infinite time (0). The <duration> is an integer of simulation time at the current resolution; you can optionally specify the resolution if you place curly braces around the specification. For example,

```
{5000 ns}
```

  sets the duration at 5000 nanoseconds regardless of the current simulator resolution.

The time range begins at the current simulation time and moves back in simulation time for the specified duration. For example,

```
vsim -wlftlim 5000
```

writes at most the last 5000ns of the current simulation to the WLF file (the current simulation resolution in this case is ns).

If used in conjunction with -wlfslim, the more restrictive of the limits will take effect.

This option may also be specified with the WLFTimeLimit variable in the *modelsim.ini* file.

The -wlfslim and -wlftlim switches were designed to help users limit WLF file sizes for long or heavily logged simulations. When small values are used for these switches, the values may be overridden by the internal granularity limits of the WLF file format. (See Limiting the WLF File Size.)

VHDL Arguments

- -absentisempty

  (optional) Causes VHDL files opened for read that target non-existent files to be treated as empty, rather than Questa SIM issuing fatal error messages.

- -accessobjdebug

  (optional) Enables logging of VHDL access type variables—both the variable value and any access object that the variable points to during the simulation. Further, it changes the default form of display-only names (such as [10001]) to a different form that you can use as input to any command that expects an object name.

  By default, logging is turned off. This means that while access variables themselves can be logged and displayed in the Questa SIM display windows, any access objects that they point to will not be logged.

  Overrides the setting for the AccessObjDebug variable in the *modelsim.ini* file.

- -foreign <attribute>

  (optional) Specifies the foreign module to load. <attribute> is a quoted string consisting of the name of a C function and a path to a shared library. For example,

  ```
  vsim -foreign "c_init for.sl"
  ```

  You can load up to ten foreign modules, for example,

  ```
  vsim -foreign "c_init for.sl" -foreign "c_init for2.so"
  ```

  Syntax for the C function and shared library is further described in the Introduction chapter of the *Questa SIM FLI Reference*.

- -noaccessobjdebug

  (optional) Disables logging of VHDL access type variables, which is the default setting. This means that while access variables themselves can be logged and displayed in the Questa SIM display windows, any access objects that they point to will not be logged.

  Overrides the setting for the AccessObjDebug variable in the *modelsim.ini* file.

- -nocollapse

  (optional) Disables the optimization of internal port map connections.

- -nofileshare

  (optional) Turns off file descriptor sharing. By default Questa SIM shares a file descriptor for all VHDL files opened for write or append that have identical names.

- -noglitch

  (optional) Disables VITAL glitch generation.

  Refer to "VHDL Simulation" for additional discussion of VITAL.

- +no_glitch_msg

  (optional) Disable VITAL glitch error messages.

- -notoggleints

  (optional) Excludes VHDL integer values from toggle coverage. Overrides the ToggleNoIntegers modelsim.ini variable default behavior of on(1).

- -novhdlvariablelogging

  (optional) This switch turns off the ability to log recursively or add process variables to the Wave or List windows. Refer to -vhdlvariable logging and VhdlVariableLogging modelsim.ini variable for more information.

- -std_input <filename>

  (optional) Specifies the file to use for the VHDL TextIO STD_INPUT file.

- -std_output <filename>

  (optional) Specifies the file to use for the VHDL TextIO STD_OUTPUT file.

- -strictvital

  (optional) Specifies to exactly match the VITAL package ordering for messages and delta cycles. Useful for eliminating delta cycle differences caused by optimizations not addressed in the VITAL LRM. Using this argument negatively impacts simulator performance.

- -togglemaxintvalues <int>

  (optional) Specifies the maximum number of VHDL integer values to record for toggle coverage. This limit variable may be changed on a per-signal basis. The default value of <int> is 100 values.

- -togglevhdlrecords | -notogglevhdlrecords

  (optional) By default, VHDL records are automatically included in code coverage metrics. The -notogglevhdlrecords argument disables the inclusion of VHDL records and -togglevhdlrecords reverses a previous -notogglevhdlrecords setting and overrides any setting made with the toggleVHDLRecords *modelsim.ini* variable.

- +transport_int_delays

  (optional) Selects transport mode with pulse control for single-source nets (one interconnect path). By default interconnect delays operate in inertial mode (pulses smaller than the delay are filtered). In transport mode, narrow pulses are propagated through interconnect delays.

  This option works for both Verilog and VITAL cells, though the destination of the interconnect must be a Verilog cell. The source may be VITAL or Verilog. This option works independently from +multisource_int_delays.

- +transport_path_delays

  (optional) Selects transport mode for path delays. By default, path delays operate in inertial mode (pulses smaller than the delay are filtered). In transport mode, narrow pulses are propagated through path delays. Note that this option affects path delays only, and not primitives. Primitives always operate in inertial delay mode.

- +typdelays

  (default) Selects the typical value in min:typ:max expressions. Has no effect if you specified the min:typ:max selection at compile time.

  If you specify the +mindelays, +typdelays, or +maxdelays flag with vopt, and specify a different flag with vsim, the simulation will be able to use the delay value based upon the flag specified with vopt. You must specify vsim -novopt to force the simulator to use the delay flag specified with vsim.

- -unattemptedimmed

  (optional) Causes any unexecuted immediate assertions to be considered in the coverage calculations for Total Coverage that are displayed in the GUI or in the coverage report. By default, any unexecuted immediate assertions are not included in the displayed calculations. This functionality can be set for all simulations through the UnattemptedImmediateAssertions *modelsim.ini* file variable.

- -vcdread <filename>

  (optional) Simulates the VHDL top-level design from the specified VCD file. This argument is included for backwards compatibility. Consider using the -vcdstim argument instead. Refer to "Simulating with Input Values from a VCD File" for more details.

- -vhdlmergepdupackage

  (optional) Turns off sharing of one package between two PDUs. Each PDU will have a separate copy of the package. This option may also be specified with the VhdlSeparatePduPackage variable in the *modelsim.ini* file.

- -vhdlseparatepdupackage

  (optional, default) Turns on sharing of packages between two or more PDUs.

- -vhdlvariablelogging

  (optional) This switch makes it possible for process variables to be logged recursively or added to the Wave and List windows (process variables can still be logged or added to the Wave and List windows explicitly with or without this switch).

  For example with this vsim switch, log -r /* will log process variables as long as vopt is specified with +acc=v and the variables are not filtered out by the WildcardFilter (via the "Variable" entry).

  You can disable this argument with -novhdlvariablelogging. Refer to -vhdlvariable logging and VhdlVariableLogging modelsim.ini variable for more information.

Note

Logging process variables inherently decreases simulation performance because of their nature. It is recommended that they not be logged or added to the Wave and List windows. However, if debugging does require logging them, then you can use this argument to minimize the performance decrease.

- -vital2.2b

  (optional) Selects SDF mapping for VITAL 2.2b (default is VITAL 2000).

Verilog Arguments

- +alt_path_delays

  (optional) Configures path delays to operate in inertial mode by default. In inertial mode, a pending output transition is cancelled when a new output transition is scheduled. The result is that an output may have no more than one pending transition at a time, and that pulses narrower than the delay are filtered. The delay is selected based on the transition from the cancelled pending value of the net to the new pending value. The +alt_path_delays option modifies the inertial mode such that a delay is based on a transition from the current output value rather than the cancelled pending value of the net. This option has no effect in transport mode (see +pulse_e/<percent> and +pulse_r/<percent>).

- -checkvifacedrivers

  (optional) Turns off/on checks for multiple-driver analysis in assignments made through virtual interfaces.

- -classdebug | -noclassdebug

  (optional) Enables/disables visibility into class instances, and includes SystemVerilog queues, dynamic arrays, and associative arrays for class and UVM debugging. You can also enable visibility into class instances by setting the ClassDebug modelsim.ini variable to 1. Refer to the classinfo find command for more information.

- -cvgbintstamp

  (optional) Records the simulation time step (timestamp) at which a covergroup bin was first covered during a simulation run. Coverage reports created from simulations in which this switch is used will contain two additional columns of data: one for timestamp value, and

another for the test name which covered the bin. See "Timestamps and UCDB Modification or Merging" for usage details.

- -cvgcollapseembeddedinstances

  Collapses embedded covergroup instances of the same type, when:

  1. option.per_instance is set to 0
  2. coverage sub-space (coverpoint bin) shape is instance independent.

  Both conditions must be met for collapsing to occur. This provides memory capacity optimization when coverage data is not required to be collected at each instance level.

- -cvghaltillbin

  (optional) Causes the simulation to halt whenever an illegal coverpoint or cross bin is hit during sampling.

- -cvgmaxrptrhscross

  (optional) Specifies maximum limit for the number of cross bin products in a coverage report and UCDB. This overrides the setting of the MaxReportRhsSVCrossProducts variable in the *modelsim.ini* file. The default value is 0.

- -cvgmergeinstances

  (optional) Sets any *type_option.merge_instances* control to 1 if there is no user-provided default assignment in the source. It overrides the SVCovergroupMergeInstancesDefault variable in the *modelsim.ini* file. This option only takes effect on covergroups that do not have any explicit assignment to *type_option.merge_instances* in user code. For more information on coverage related to merge_instances, see "IEEE Std 1800-2009 Option Behavior".

- -cvgopt[=[+|-]<mode>[,[+|-]<mode>]...]

  (optional) Enables the covergroup optimization modes. Valid modes are:

  minhitcnt — Stops sampling coverpoint/cross when it is fully covered.

- -cvgperinstance

  (optional) Sets the *option.per_instance* control in all covergroup declarations to 1, to enable the debugging of covergroups, overriding the user's setting. This option takes effect on all coverage objects in the design, regardless of whether you explicitly assign a value to *option.per_instance*. For more information on coverage related to per_instance, see "IEEE Std 1800-2009 Option Behavior".

- -cvgprecollect <ucdb_file>

  (optional) Specifies a UCDB file to be used for optimization control in the current simulation. This switch can occur multiple times on the same command line. (See Controlling Optimization with a UCDB File)

- -cvgsingledefaultbin

  (optional) Collapses a default array bin to a scalar bin. By default, a default array bin is modeled as a sparse array and the sub-bins are reported only when there are one or more samples.

- -cvgzwnocollect

  (optional) Disables coverage collection for any coverage item (coverpoint or cross or the entire covergroup) when 0 is assigned as its *option.weight*. Item will not be displayed in any coverage report or contribute to any coverage score computation.

- +delayed_timing_checks

  (optional) Causes timing checks to be performed on the delayed versions of input ports (used when there are negative timing check limits). By default, Questa SIM automatically detects and applies +delayed_timing_checks to cells with negative timing checks. To turn off this feature, specify +no_autodtc with vsim.

- -dpicppinstall <[gcc | g++] version>

  (optional) Specifies the version of the desired GNU compiler supported and distributed by Mentor Graphics for the DPI exportwrapper compilation.

  <[gcc | g++] version> — The version number of the GNU compiler to use. For example:

  ```
  vsim -dpicppinstall 4.5.0
  ```

- -dpicpppath <pathname>

  (optional) Specifies an explicit location to a gcc compiler for use with automatically generated DPI exportwrappers. Ensures that the argument points directly to the compiler executable. This overrides the DpiCppPath variable in the *modelsim.ini* file.

- -dpiforceheader

  (optional) Forces the generation of a DPI header file even if it will be empty of function prototypes.

- -dpiheader

  (optional) Generates a header file that may then be included in C source code for DPI import functions. Simulation quits after header file is generated. Refer to "DPI Use Flow" for additional information.

- -dpilib <libname>

  (optional) Specifies the design library name that contains DPI exports and automatically compiled object files. If the -dpilib switch is not set, vsim loads export symbols from all libraries accessible via vsim options -L, -Lf, and -lib. Multiple occurrences of -dpilib are supported.

- -dpioutoftheblue 0 | 1 | 2

  (optional) Instructs vsim to allow DPI out-of-the-blue calls from C functions. The C functions must not be declared as import tasks or functions.

0 — Support for DPI out-of-the-blue calls is disabled.

1 — Support for DPI out-of-the-blue calls is enabled, but debugging support is not available.

2 — Support for DPI out-of-the-blue calls is enabled with debugging support for a SystemC thread.

Debugging support for DPI out-of-the-blue calls from a SystemC method requires two vsim arguments entered together at the command line: -dpioutoftheblue 2 and -scdpidebug. Refer to -scdpidebug for more information.

Related *modelsim.ini* file variable is DpiOutOfTheBlue.

- -gconrun | -nogconrun

  (optional) Enables/disables garbage collector execution after each simulation run command completes.

- -gconstep | -nogconstep

  (optional) Enables/disables garbage collector execution after each step when stepping through your simulation.

- -gcthreshold <n>

  (optional) Sets the maximum amount of memory in megabytes allocated for storage of class objects before the garbage collector runs to delete unreferenced objects.

  <n> — Any positive integer where <n> is the number of megabytes. The default size is 100 megabytes.

  Related *modelsim.ini* file variables are GCThreshold and GCThresholdClassDebug.

- -hazards

  (optional) Enables event order hazard checking in Verilog modules (Verilog only). You must also specify this argument when you compile your design with vlog. Refer to "Hazard Detection" for more details.

Note

Using -hazards implicitly enables the -compat argument. As a result, using this argument may affect your simulation results.

- -ignoreinilibs

  (optional) Ignore the libraries specified in the LibrarySearchPath variable in the vsim section of the *modelsim.ini* file.

- +initmem+<seed>

  (optional) Specifies the seed value to be used by random initialization for Verilog designs. Random initialization (of only 0 or 1) occurs at runtime for memories compiled by vlog with the +initmem option without specifying a modifier (+{0 | 1 | X | Z}).

If no +initmem is present on the vsim command line, a random seed of 0 is used during initialization.

+<seed> — any signed 32-bit integer (-2147483648 to +2147483647).

- +initreg+<seed>

(optional) Specifies the seed value to be used by random initialization for Verilog designs. Random initialization (of only 0 or 1) occurs at runtime for registers compiled by vlog with the +initreg option without specifying a modifier (+{0 | 1 | X | Z}).

If no +initreg is present on the vsim command line, a random seed of 0 is used during initialization.

+<seed> — any signed 32-bit integer (-2147483648 to +2147483647).

- -initreport <filename>

(optional) Saves a report of the initial seed values generated by vlog +initmem/+initreg or vopt +initmem/+initreg to a file.

<filename> — (required) A string specifying the name for the report.

Output is displayed in the following format:

```
scope:
<type> <name> <N : N> <value>
```

where

scope is the relative path to the scope for the registers and memories that follow.

<type> — is the type of object modified by +initmem or +initreg.

m — indicates a memory
r — indicates a register
u — indicates an initialized UDP

<name> — is the name of the object.

<N : N> — is the dimension of the register or memory.

<value> — is the seed value set by +initmem or +initreg.

For example:

```
scope: /tb_qctest/i_qctest_top/i_qctest_256k/i_kfla
m ram_model [1:0] : [0:262143] '0
r dout [1:0] '0
r ua [0:0] '0
```

- +maxdelays

(optional) Selects the maximum value in min:typ:max expressions. The default is the typical value. Has no effect if you specified the min:typ:max selection at compile time.

If you specify the +mindelays, +typdelays, or +maxdelays flag with vopt, and specify a different flag with vsim, the simulation will be able to use the delay value based upon the

flag specified with vopt. You must specify vsim -novopt to force the simulator to use the delay flag specified with vsim.

- +mindelays

  (optional) Selects the minimum value in min:typ:max expressions. The default is the typical value. Has no effect if you specified the min:typ:max selection at compile time.

  If you specify the +mindelays, +typdelays, or +maxdelays flag with vopt, and specify a different flag with vsim, the simulation will be able to use the delay value based upon the flag specified with vopt. You must specify vsim -novopt to force the simulator to use the delay flag specified with vsim.

- -noappendclose

  (optional) Simulator will ***not*** immediately close files opened in APPEND mode. Designed to override the AppendClose *modelsim.ini* variable when it is set to one (On). Subsequent calls to file_open in APPEND mode will therefore not require operating system interaction, resulting in faster performance.

- +no_autodtc

  (optional) Turns off auto-detection of optimized cells with negative timing checks and auto-application of +delayed_timing_checks to those cells.

- +no_cancelled_e_msg

  (optional) Disables negative pulse warning messages. By default vsim issues a warning and then filters negative pulses on specify path delays. You can drive an X for a negative pulse using +show_cancelled_e.

- +no_neg_tchk

  (optional) Disables negative timing check limits by setting them to zero. By default negative timing check limits are enabled. This is just the opposite of Verilog-XL, where negative timing check limits are disabled by default, and they are enabled with the +neg_tchk option.

- +no_notifier

  (optional) Disables the toggling of the notifier register argument of all timing check system tasks. By default, the notifier is toggled when there is a timing check violation, and the notifier usually causes a UDP to propagate an X. This argument suppresses X propagation on timing violations for the entire design.

  You can suppress X propagation on individual instances using the tcheck_set command.

- +no_path_edge

  (optional) Causes Questa SIM to ignore the input edge specified in a path delay. The result of this argument is that all edges on the input are considered when selecting the output delay. Verilog-XL always ignores the input edges on path delays.

- +no_pulse_msg

  (optional) Disables the warning message for specify path pulse errors. A path pulse error occurs when a pulse propagated through a path delay falls between the pulse rejection limit

and pulse error limit set with the +pulse_r and +pulse_e options. A path pulse error results in a warning message, and the pulse is propagated as an X. The +no_pulse_msg option disables the warning message, but the X is still propagated.

- -no_risefall_delaynets

  (optional) Disables the rise/fall delay net delay negative timing check algorithm. This argument is provided to return Questa SIM to its pre-6.0 behavior where violation regions must overlap in order to find a delay net solution. In 6.0 versions and later, Questa SIM uses separate rise/fall delays, so violation regions need not overlap for a delay solution to be found.

- +no_show_cancelled_e

  (optional) Filters negative pulses on specify path delays so they don't show on the output. Default. Use +show_cancelled_e to drive a pulse error state.

- +no_tchk_msg

  (optional) Disables error messages issued by timing check system tasks when timing check violations occur. Notifier registers are still toggled and may result in the propagation of Xs for timing check violations.

  You can disable individual messages using the tcheck_set command.

- -nocvg

  (optional) Disables covergroup object construction. It also removes the ability to run builtin covergroup methods during simulation. If you set this switch, simulation will fail if there is any hierarchical access to covergroup option variables. This runtime switch is intended to assist in determining if simulation performance is being impacted by covergroup explosion; however in doing so, it fully disables the SystemVerilog functional coverage feature. Therefore, this switch is not recommended for production purposes. Covergroup does not require the svverification license when -nocvg is used in the simulation.

- -nocvgmergeinstances

  (optional) Sets any *option.merge_instances* control to 0 if there is no user-provided default assignment in the source. It overrides the SVCovergroupMergeInstancesDefault variable in the *modelsim.ini* file when the variable is set to 1. This option only takes effect on covergroups that do not have any explicit assignment to *option.merge_instances* in user code.

- -nodpiexports

  (optional) Instructs Questa SIM to not generate C wrapper code for DPI export task and function routines found at elaboration time. More specifically, the command does not generate the *exportwrapper.so* shared object file.

  For a description on when you should use this argument, refer to the section "Deprecated Legacy DPI Flows" in the User's Manual.

- -noexcludehiz

  (optional) Instructs Questa SIM to include truth table rows that contain Hi-Z states in the coverage count. Without this argument, these rows are automatically excluded.

- -noexcludeternary

  (optional) Disables the automatic exclusion of UCDB coverage data rows resulting from ternary expressions for the entire design. Normal operation for code coverage is to include rows corresponding to the case where two data inputs are the same, and the select input is a “don’t care”. To disable this automatic exclusion for a specified design unit only, use “vlog -noexcludeternary <design_unit>” instead.

- -noimplicitcoverpoint

  (optional) Avoids collecting coverage for implicit coverpoints. By default, coverage is collected for coverpoints implicitly defined in a cross.

- +nosdferror

  (optional) Errors issued by the SDF annotator while loading the design prevent the simulation from continuing, whereas warnings do not. Changes SDF errors to warnings so that the simulation can continue.

- +nosdfwarn

  (optional) Disables warnings from the SDF annotator.

- +nospecify

  (optional) Disables specify path delays and timing checks.

- +nowarnBSOB

  (optional) Disables run-time warning messages for bit-selects in initial blocks that are out of bounds.

- +nowarn<CODE | number>

  (optional) Disables warning messages in the category specified by a warning code or number. Warnings that can be disabled include the code name in square brackets in the warning message. For example:

  ```
  ** Warning: (vsim-3017) test.v(2): [TFMPC] - Too few port
  connections. Expected <m>, found <n>.
  ```

  The warning code for this example is TFMPC, and the warning number is 3017. Therefore, this warning message can be disabled with +nowarnTFMPC or +nowarn3017.

- +ntc_warn

  (optional) Enables warning messages from the negative timing constraint algorithm. By default, these warnings are disabled.

  This algorithm attempts to find a set of delays for the timing check delayed net arguments such that all negative limits can be converted to non-negative limits with respect to the delayed nets. If there is no solution for this set of limits, then the algorithm sets one of the

negative limits to zero and recalculates the delays. This process is repeated until a solution is found. A warning message is issued for each negative limit set to zero.

- +ntcnotchks

  (optional) Instructs vsim to not simulate timing checks but still consider negative timing check limits for the calculation of delayed input delays.

- -oldvhdlforgennames

  (optional) Enables the use of a previous style of naming in VHDL for … generate statement iteration names in the design hierarchy. The previous style is controlled by the value of the GenerateFormat value. The default behavior is to use the current style names, which is described in "Naming Behavior of VHDL For Generate Blocks" This argument duplicates the function of the OldVhdlForGenNames variable in modelsim.ini and will override the setting of that variable if it specifies the current style.

- -onfinish ask | stop | exit | final

  (optional) Customizes the simulator shutdown behavior when it encounters $finish or sc_stop() in the design:

  - ask —
    - In batch mode, the simulation exits.
    - In GUI mode, a dialog box pops up and asks for user confirmation on whether to quit the simulation.
  - stop — stops simulation and leaves the simulation kernel running
  - exit — exits out of the simulation without a prompt
  - final — executes all final blocks then exits the simulation

  By default, the simulator exits in batch mode; prompts you in GUI mode. Edit the OnFinish variable in the *modelsim.ini* file to set the default operation of $finish.

- -pedanticerrors

  (optional) Forces display of an error message (rather than a warning) on a variety of conditions.

  You can view a complete list of errors by executing the command:

  ```
  verror -kind vsim -pedanticerrors
  ```

- -permit_unmatched_virtual_intf

  (optional) Permits vsim to elaborate designs containing virtual interface declarations for which no actualy interface instances exist which are compatible with that declaration. Such virtual interface declarations are considerted "unmatched" as there is no matching or compatible interface instance. By default, unmatched virtual interfaces prevent vsim from elaborating the design. For further information on this design situation, see "Unmatched Virtual Interface Declarations".

- -pli "<object list>"

  (optional) Loads a space-separated list of PLI shared objects. The list must be quoted if it contains more than one object. This is an alternative to specifying PLI objects in the Veriuser entry in the *modelsim.ini* file, refer to modelsim.ini Variables. You can use environment variables as part of the path.

- -plicompatdefault [latest | 2009 | 2005 | 2001]

  (optional) Specifies the VPI object model behavior within vsim. This switch applies globally, not to individual libraries.

  latest — This is equivalent to the "2009" argument. This is the default behavior if you do not specify this switch or if you specify the switch without an argument.

  2009 — Instructs vsim to use the object models as defined in IEEE Std P1800-2009 (unapproved draft standard). You can also use "09" as an alias.

  2005 — Instructs vsim to use the object models as defined in IEEE Std 1800-2005 and IEEE Std 1364-2005. You can also use "05" as an alias.

  2001 — Instructs vsim to use the object models as defined in IEEE Std 1364-2001. When you specify this argument, SystemVerilog objects will not be accessible. You can also use "01" as an alias.

  You can also control this behavior with the PliCompatDefault variable in the modelsim.ini file, where the -plicompatdefault argument will override the PliCompatDefault variable.

  You should note that there are a few cases where the 2005 VPI object model is incompatible with the 2001 model, which is inherent in the specifications.

  Refer to the appendix "Verilog Interfaces to C" in the User's Manual for more information.

- +<plusarg>

  (optional) Arguments preceded with "+" are accessible by the Verilog PLI routine mc_scan_plusargs(). All plusarg argument values can be overridden in -restore mode and will take effect when simulation resumes after restoring the design.

- +pulse_e/<percent>

  (optional) Controls how pulses are propagated through specify path delays, where <percent> is a number between 0 and 100 that specifies the error limit as a percentage of the path delay.

  A pulse greater than or equal to the error limit propagates to the output in transport mode (transport mode allows multiple pending transitions on an output). A pulse less than the error limit and greater than or equal to the rejection limit (see +pulse_r/<percent>) propagates to the output as an X. If the rejection limit is not specified, then it defaults to the error limit. For example, consider a path delay of 10 along with a +pulse_e/80 option. The error limit is 80% of 10 and the rejection limit defaults to 80% of 10. This results in the propagation of pulses greater than or equal to 8, while all other pulses are filtered. Note that you can force specify path delays to operate in transport mode by using the +pulse_e/0 option.

- +pulse_e_style_ondetect

  (optional) Selects the "on detect" style of propagating pulse errors (see +pulse_e). A pulse error propagates to the output as an X, and the "on detect" style is to schedule the X immediately, as soon as it has been detected that a pulse error has occurred. "on event" style is the default for propagating pulse errors (see +pulse_e_style_onevent).

- +pulse_e_style_onevent

  (optional) Selects the "on event" style of propagating pulse errors (see +pulse_e). Default. A pulse error propagates to the output as an X, and the "on event" style is to schedule the X to occur at the same time and for the same duration that the pulse would have occurred if it had propagated through normally.

- +pulse_r/<percent>

  (optional) Controls how pulses are propagated through specify path delays, where <percent> is a number between 0 and 100 that specifies the rejection limit as a percentage of the path delay.

  A pulse less than the rejection limit is suppressed from propagating to the output. If the error limit is not specified by +pulse_e then it defaults to the rejection limit.

- +sdf_nocheck_celltype

  (optional) Disables the error check a for mismatch between the CELLTYPE name in the SDF file and the module or primitive name for the CELL instance. It is an error if the names do not match.

- +show_cancelled_e

  (optional) Drives a pulse error state (’X’) for the duration of a negative pulse on a specify path delay. By default Questa SIM filters negative pulses.

- -showlibsearchpath

  (optional) Returns to the transcript all libraries that will be searched for precompiled modules.

- -solvebeforeerrorseverity=<value>

  (optional) Modifies the severity of suppressible index, out-of-bounds, null-dereference, solve/before constraint errors. You can edit the SolveBeforeErrorSeverity variable in the *modelsim.ini* file to set a permanent default value.

  Valid values:

  0 — No error

  1 — Warning

  2 — Error

  3 — Failure (default)

  4 — Fatal

- -solveengine {auto | bdd | act}

  (optional) Selects a solver “engine” to be used for constrained random generation. Overrides the setting for the SolveEngine variable in the *modelsim.ini* file. The following values for this argument are case-insensitive (for example, you can specify BDD or bdd).

  auto — (default) automatically selects the best engine for the current randomization operation.

  bdd — evaluates all randomization operations using the Binary Decision Diagram (BDD) solver engine. The BDD-based solver is usually more efficient with sets of constraints involving certain bit-wise logical relationships.

  act — evaluates all randomization operations using the Arithmetic Constraint Tree (ACT) solver engine. The ACT-based solver can exhibit superior performance with constraints that entail heavy use of arithmetic operators (+, -, *, /).

- -solvefaildebug=<value>

  (optional) Enables SystemVerilog randomize() failure debugging. Whenever a randomize() failure is detected during simulation, Questa SIM will display the minimum set of constraints that caused the randomize() call to fail.

  The simulator also prints to the transcript a simplified testcase of the failing randomization scenario. This testcase contains solve/before constraints, so you can use it to reproduce the failure or as an aid in debugging the constraints. Variable/field names and file names of constraints in the protected region are hidden.

  Valid values:

  0 - disable solvefaildebug

  1 - basic debug (provides a testcase and prints contradicting constraints with no performance penalty)

  2 - enhanced debug (provides constraints in more original form with a runtime performance penalty)

  If no value is specified, basic debug will be enabled.

- -solvefailtestcase[=<filename>]

  (optional) Generates a simplified testcase when a randomize() failure is encountered and outputs the testcase to the specified file. If no filename is specified, the testcase will be output to the Transcript. Testcases for 'no-solution' failures will only be produced if '-solvefaildebug' is enabled.

- -solveprofile

  (optional) Enables basic solver profiling, which records performance information related to SystemVerilog randomize() calls invoked during simulation. When solver profiling is enabled, the solver report (“write report -solver” command) will include profile data.

- -solverev <version>

  (optional) Specifies random sequence generation compatibility with a prior letter release for the SystemVerilog solver. (It does not apply to the SystemC/SCV solver.) This argument is

used to get the same random sequences during simulation as a prior letter release. The <version> is a string of a release number and letter, such as 6.2a. (-solverev 6.2a). Only prior letter releases (within same number release) are allowed. For example, in 6.2b you can specify "-solverev 6.2" or "-solverev 6.2a", but cannot specify "-solverev 6.1f". See also, the *modelsim.ini* "SolveRev" variable.

- -solvetimeout <val>

  (optional) Specify the solver timeout threshold (in seconds), to improve the handling of randomize() timeouts. A randomize() call will fail if the CPU time required to evaluate any randset exceeds the specified timeout. Default: 500. A value of 0 disables the timeout feature. See also, the *modelsim.ini* "SolveTimeout" variable.

- -solveverbose

  (optional) Reports information about randomize() call processing.

- -sv_lib <shared_obj>

  (required for use with DPI import libraries) Specifies the name of the DPI shared object with no extension. Refer to "DPI Use Flow" for additional information.

- -sv_liblist <filename>

  (optional) Specifies the name of a bootstrap file containing names of DPI shared objects (libraries) to be loaded. Refer to "DPI File Loading" for format information.

- -sv_root <dirname>

  (optional) Specifies the directory name to be used as the prefix for DPI shared object lookups.

- -sv_seed <integer> | random

  (optional) Seeds the root random number generator for SystemVerilog threads with either a user-specified integer or a random number generated by Questa SIM. Also seeds the shuffle() array ordering operator. The integer may be any signed 32-bit integer (-2147483648 to +2147483647). Integers outside the value range will be assigned either the minimum or maximum value. Edit the Sv_Seed variable in the *modelsim.ini* file to set a permanent default. Refer to "Seeding the Random Number Generator (RNG)" for more information.

- -svext=[+|-]<extension>[,[+|-]<extension>]…

  (optional) Enables SystemVerilog language extensions through a comma-separated list of arguments.

  [+ | -] — controls activation of the *extension*. Remember, arguments on the command line control precedence. Any settings to this switch will override your settings of the SvExtensions *modelsim.ini* variable.

  + — activates the *extension*.

  - — deactivates the *extension*.

If you do not specify either a “+” or “-”, the command assumes you are activating the specified *extension*.

<extension> —

cfce — Error message will be generated if $cast is used as a function and the casting operation fails.

iddp — Specifies that the DPI disable protocol check be ignored. For example, the following fatal error will be downgraded to a suppressible warning: *** Fatal: (vsim-3765) The exported task or function 'block20' has been called from within a disabled context. This is illegal.*

nonrandstab — When enabled, any allocaiton of a new “non random” class instance will not consume a random number from the current thread context. A class is considered “non random” when it, or its base classes, does not declare any rand or randc fields, or it does not invoke class::randomize() anywhere.

expdfmt — Interprets format specifiers in string variables passed to $display as format specifiers, not normal strings. For $sformatf, arguments outnumbering the number of format specifiers will be appended to the return string.

- -togglefixedsizearray | -notogglefixedsizearray

  (optional) The -togglefixedsizearray argument includes SystemVerilog unpacked fixed-size arrays, as well as VHDL multi-dimensional arrays and arrays-of-arrays, in toggle coverage. By default, these types of arrays are not considered for toggle coverage. The -togglefixedsizearray argument overrides the ToggleFixedSizeArray modelsim.ini variable default setting of off (0).

- -togglemaxfixedsizearray <int>

  (optional) Specifies the maximum size for the SystemVerilog unpacked real type fixed-size arrays, as well as VHDL multi-dimensional arrays and arrays-of-arrays, that are collected for toggle coverage. By default, large fixed-sized arrays (>1024 elements) are not included in toggle coverage, even when the -togglefixedsizearray option is used, as this can have an adverse impact on simulation performance. Use the ToggleMaxFixedSizeArray *modelsim.ini* variable to control this limit.

- -togglemaxrealvalues <int>

  (optional) Specifies the maximum number of SystemVerilog real values to record for toggle coverage of a given signal. This limit variable may be changed on a per-signal basis. The default value of 100 values can be modified by editing the ToggleMaxRealValues *modelsim.ini* variable.

- -togglepackedasvec

  (optional) Specifies that SystemVerilog packed structures and multi-d arrays are treated as flattened vectors for toggle coverage. Overrides the TogglePackedAsVec *modelsim.ini* variable default setting of off (0).

- -toggleportsonly

  (optional) Enables only ports for inclusion in toggle coverage numbers; no internal signals are included in coverage metrics. If you want to see coverage of all ports in the design, use the “vopt +acc=p” command — but note that there could be a significant performance penalty due to the fact that inlining is turned off. You can selectively enable toggle coverage on specific ports by using the “vopt +acc=p+<selection>” command. Overrides any global value set by the TogglePortsOnly *modelsim.ini* variable.

- -togglevlogenumbits

  (optional) Specifies that SystemVerilog enum types are treated as reg-vectors for toggle coverage. Overrides the default setting of the ToggleVlogEnumBits variable in *modelsim.ini*, which is off(0).

- -togglevlogints | -notogglevlogints

  (optional) By default, SystemVerilog integer types (shortint, int, longint, byte, integer and time) are treated as reg-vectors, and counts are kept for each bit. The -notogglevlogints argument excludes these type from coverage, overriding the default setting of the ToggleVlogIntegers variable in *modelsim.ini*, which is on(1). The -togglevlogints argument is used to enable coverage after it has been disabled.

- -togglevlogreal | -notogglevlogreal

  (optional) The -togglevlogreal argument includes Verilog real value types in toggle coverage. Overrides the default setting of the ToggleVlogReal variable in *modelsim.ini*, which is off (0).

- -trace_dpi <val>

  (optional) Sets the tracing level for DPI-C. By default, tracing for DPI-C import/export calls is disabled (0). Valid <val> settings are:

  1 — Enables all tracing

  0 — Disables all tracing

  i — Enables the tracing of DPI import calls only

  e — Enables the tracing of DPI export calls only

  a — Enables the tracing of DPI import/export call arguments only

- +transport_int_delays

  (optional) Selects transport mode with pulse control for single-source nets (one interconnect path). By default interconnect delays operate in inertial mode (pulses smaller than the delay are filtered). In transport mode, narrow pulses are propagated through interconnect delays.

  This option works for both Verilog and VITAL cells, though the destination of the interconnect must be a Verilog cell. The source may be VITAL or Verilog. This option works independently from +multisource_int_delays.

- +transport_path_delays

  (optional) Selects transport mode for path delays. By default, path delays operate in inertial mode (pulses smaller than the delay are filtered). In transport mode, narrow pulses are propagated through path delays. Note that this option affects path delays only, and not primitives. Primitives always operate in inertial delay mode.

- +typdelays

  (default) Selects the typical value in min:typ:max expressions. Has no effect if you specified the min:typ:max selection at compile time.

  If you specify the +mindelays, +typdelays, or +maxdelays flag with vopt, and specify a different flag with vsim, the simulation will be able to use the delay value based upon the flag specified with vopt. You must specify vsim -novopt to force the simulator to use the delay flag specified with vsim.

- -unattemptedimmed

  (optional) Causes any unexecuted immediate assertions to be considered in the coverage calculations for Total Coverage that are displayed in the GUI or in the coverage report. By default, any unexecuted immediate assertions are not included in the displayed calculations. This functionality can be set for all simulations through the UnattemptedImmediateAssertions *modelsim.ini* file variable.

- -v2k_int_delays

  (optional) Causes interconnect delays to be visible at the load module port per the IEEE 1364-2001 spec. By default Questa SIM annotates INTERCONNECT delays in a manner compatible with Verilog-XL. If you have $sdf_annotate() calls in your design that are not getting executed, add the Verilog task $sdf_done() after your last $sdf_annotate() to remove any zero-delay MIPDs that may have been created. May be used in tandem with the +multisource_int_delays argument (see above).

  Refer to sdfcom for SDF compilation information.

- -wrealdefaultzero

  (optional) For nets declared as wreal, sets the default value for an undriven wreal net to zero (0).

- -wreal_resolution <val>

  (optional) For nets declared as wreal, selects the resolution function for computing the effective value for multiple drivers on the net. Conflicting values on a network are passed through the selected resolution function. Valid enum values for val are:

  DEFAULT — Single active driver only

  4STATE — Verilog 4-state resolution

  SUM — Sum of all driver values

  AVG — Average of all driver values

  MIN — Smallest of all driver values

MAX — Largest of all driver values

SystemC Arguments

- -noscmainscopename

  (optional) When design units are instantiated under sc_main, Questa SIM creates a scope called sc_main which gets appended to the hierarchical path returned by the name() function, which renders the design incompatible with OSCI. The -noscmainscopename option strips out the sc_main scope name from the output of the name() function.

- -scdpidebug

  (optional) Enables DPI debug single-stepping across SystemC-SystemVerilog call boundaries for SystemVerilog breakpoints placed inside an export function call that was initiated from an SC_METHOD.

  Refer to the sections "Setting Breakpoints" and "Stepping in C Debug" for more information.

  Turns on debugging support for DPI out-of-the-blue calls from a SystemC method when combined with the vsim argument -dpioutoftheblue. Refer to -dpioutoftheblue for more information.

- -sclib <library>

  (optional) Specifies the design library where the SystemC shared library is created. By default, the SystemC shared library is created in the logical work library. This option is only necessary when the shared library is compiled in a design library other than the logical work directory (via sccom -link -work <lib>). For more information on the sccom -link and -work arguments, see sccom.

- -scstacksize <value>

  (optional) Set global stack size for all SystemC threads in the design to the value supplied.

  <value> — an integer number followed by the size unit that can be Kb(Kilo-byte), Mb(Mega-byte) or Gb(Giga-byte). Examples:

  ```
  vsim -scstacksize '1000 Kb'
  vsim -scstacksize '1 Mb'
  vsim -scstacksize '1 Gb'
  ```

  You can also set the stack size for all SystemC threads using the *modelsim.ini* file variable named 'ScStackSize'.

- -sc_arg <string> ...

  (optional) Specifies a string representing a startup argument which is subsequently accessible from within SystemC via the sc_argc() and sc_argv() functions (refer to "Accessing Command-Line Arguments in SystemC-22".

  If multiple SystemC startup arguments are specified, each must have a separate -sc_arg argument. SystemC startup arguments returned via sc_argv() are in the order in which they appear on the command line. White space within the <string> will not be treated specially,

and the string, white space and all, will be accessible as a single string among the strings returned by sc_argv().

Visualizer Debug Environment Arguments

Invokes the VSIM simulator and creates the simulation database files required to log data for post-simulation debugging with Mentor Visualizer Debug Environment. The -qwavedb argument records simulation data for post-simulation debugging. Optional arguments to -qwavedb log specific simulation data to the database file.

You can specify -qwavedb and -load_elab <filename> together on the vsim command line to load a previously generated elaboration file.

Arguments must be specified as a string without spaces.

Postsimulation Debug

vsim <vsim_args> **-qwavedb[=+<option>[+<option>+...]]** [+class_memleak_threshold+<n>] [+max_bits_per_wave+<n>] [+qwavedb_include_cells] [+qwavedb_log_vhdl_vars] [+qwavedb_dump_start+<start_time>] [+qwavedb_dump_stop+<stop_time>]

Arguments

- **-qwavedb**[=+<*option*>[+<*option*>+...]]

  (required) — Creates the simulation database file required to run Questa SIM. Optional arguments to -qwavedb log specific simulation data to the database file. Switches must be specified as a string without spaces.

Note

Specifying **vsim** -qwavedb disables WLF file generation and is incompatible with all WLF switches and commands.

Where <*option*> is one of the following:

+assertion[=pass]

(optional) Logs failed assertions by default.

pass — (optional) Log passing and failing assertions.

+assocarray

(optional) Log associative arrays.

+class[=<*class_filter_file*>]

(optional) Log classes. When specified alone, logs all classes in the design.

<*class_filter_file*> — (optional) Specifies a user supplied file containing classes to log. Refer to Class Filter File Format in *Mentor Visualizer Debug Environment* for more information.

+displaymsgmode[=log | both | tran]

(optional) Log $display, $strobe, $monitor, and so forth, System Task messages. Includes “both” and “tran” where the default is “both”.

+dynamic

(optional) Log all dynamic objects. For example, arrays, queues, classes, transactions, messages, and so forth.

+dynarray[=<*n*>]

(optional) Log dynamic arrays.

=<*n*> — Specifies the depth to which dynamic arrays are logged.

+glitch

(optional) Enables recording of multiple value changes during a given simulation time point into the *qwavedb* file. Normally, a single value change is recorded for any object at any time. Under the glitch option, if any object changes its value multiple times during a given simulation time point, all the different values are recorded as long as the changes occur during different delta cycles within that simulation time point. The delta cycle numbers are recorded as well.

The command line switch -showglitch must be specified with the **visualizer** command to display the logged glitches in waveforms. Refer to Mentor Visualizer Debug Environment documentation for more information.

Note

Use of this option significantly increases the time it takes to log waveforms while also increasing the size of the generated waveform (*qwavedb*) file significantly.

+memory[=<*n*>]

(optional) Logs memories.

=<*n*> — (optional) Specifies a maximum size of memory to log.

+msgmode=log | both | tran

(optional) Log all messages including “both” and “tran” where the default is “both.”

+queue

(optional) Log all queues.

+report[=class]

(optional) Generate a Dump Summary report of everything logged to the simulation database file.

=class — (optional) Specifies a Dump Summary report of all classes logged to the simulation database file.

+rollover[=<*m*>[,<*n*>]]

(optional) Logs simulation data to a circular buffer and limits simulation database file size to 5000 MB by default. File names are named sequentially, for example, *qwave.db_1*, *qwave.db_2*, and so forth.

You can also log simulation data to multiple sequential files and specify the number of files. The +rollover option is implemented as a circular buffer with data written to

each file in turn once the maximum file size is reached for the current log file. When the maximum number of files is reached, data recording rolls over to the first file.

Prints a record of the files with simulation data and the simulation time span in the file.

<*m*> — (optional) Limits file size to <m> megabytes maximum.

<*n*> — (optional) Specifies the number of log files used to save simulation data.

+signal[=<*signal_file*>]

(optional) Enables RTL (DUT) full wave dumping to the simulation database.

=<*signal_file*> — (optional) A user supplied file specifying RTL depth and scopes to be dumped. Refer to Signal File Format in *Mentor Visualizer Debug Environment* for more information.

+transactions[=[uvm | vip | <*transaction_filter_file*>]]

(optional) Log all transactions.

uvm — (optional) Log UVM transactions.

vip — (optional) Log VIP transactions.

<*transaction_filter_file*> — (optional) Specifies a user supplied file with transaction filters. Refer to Transaction Filter File Format in *Mentor Visualizer Debug Environment* for more information.

+uvm_schematic

(optional) Logs UVM Schematic objects.

+wavefile[= | +]<*filename*>

(optional) Logs simulation data to a specified file. .

<*filename*> — A user specified string.

- +class_memleak_threshold+<*n*>

  (optional) Reports the classes that have greater than 1000 un-deleted instances. You can change the threshold by specifying +<*n*>.

  <n> — Any positive integer.

- +max_bits_per_wave+<*n*>

  (optional) Increases the number of bits logged for each wave signal where the default is to log waves up to 1024 bits.

  <*n*> — Any positive integer.

- +qwavedb_include_cells

  (optional) By default, variables inside a cell are not logged. This enables logging inside cells

- +qwavedb_log_vhdl_vars

  (optional) Enables logging of VHDL process variables.

- +qwavedb_dump_start+<*start_time*>

  (optional) Start dumping waveforms at specified start time.

  <*start_time*> — Any positive integer. The time unit is the Simulator Resolution Limit.

- +qwavedb_dump_stop+<*stop_time*>

  (optional) Stop dumping waveforms at a specified simulation time.

  <*stop_time*> — Any positive integer. The time unit is the Simulator Resolution Limit.

Object Arguments

The object arguments may be a [<library_name>].<design_unit>, an *.mpf* file, a *.wlf* file, or a text file. Multiple design units may be specified for Verilog modules and mixed VHDL/Verilog configurations.

- <library_name>.<design_unit>

  (optional) Specifies a library and associated design unit; multiple library/design unit specifications can be made. If no library is specified, the work library is used. You cannot use the wildcard * for this argument. Environment variables can be used. <design_unit> may be one of the following:

<configuration>	Specifies the VHDL configuration to simulate.
<module> …	(optional) Specifies the name of one or more top-level Verilog modules to be simulated.
<entity> [(<architecture>)]	(optional) Specifies the name of the top-level VHDL entity to be simulated. The entity may have an architecture optionally specified; if omitted the last architecture compiled for the specified entity is simulated. An entity is not valid if a configuration is specified.[1]
<optimized_design_name>	(optional) Specifies the name(s) of the optimized design(s). See the vopt command.

1. Most UNIX shells require arguments containing () to be single-quoted to prevent special parsing by the shell. See the examples below.

  Multiple optimized top modules can be specified as a space separated list. If all design modules are optimized they will be elaborated as individual optimized designs. Where top modules are a combination of optimized and un-optimized design units, the un-optimized design units will not be optimized and will be elaborated as un-optimized top modules. If all of the top modules are un-optimized, they will be optimized during elaboration unless -novopt is specified. Support for elaboration of multiple top design units for Power Aware and Schematic debugging is supported only for designs with all design units are un-optimized.

- <MPF_file_name>

  (optional) Opens the specified project.

- <WLF_file_name>

  (optional) Opens the specified dataset. When you open a WLF file using the following command:

  vsim test.wlf

  The default behavior is to not automatically load any signals into the Wave window. You can change this behavior, such that the Wave window contains all signals in the design, by setting the preference PrefWave(OpenLogAutoAddWave) to 1 (true).

- <text_file_name>

  (optional) Opens the specified text file in a Source window.

Examples

- Invoke vsim on the entity *cpu* and assigns values to the generic parameters *edge* and *VCC*.

  ```
  vsim -gedge=’"low high"’ -gVCC=4.75 cpu
  ```

  If working within the Questa SIM GUI, you would enter the command as follows:

  ```
  vsim {-gedge="low high"} -gVCC=4.75 cpu
  ```

  Instruct Questa SIM to view the results of a previous simulation run stored in the WLF file *sim2.wlf*. The simulation is displayed as a dataset named *test*. Use the -wlf option to specify the name of the WLF file to create if you plan to create many files for later viewing.

  ```
  vsim -view test=sim2.wlf
  ```

  For example:

  ```
  vsim -wlf my_design.i01 my_asic structure
  vsim -wlf my_design.i02 my_asic structure
  ```

  Annotate instance */top/u1* using the minimum timing from the SDF file *myasic.sdf*.

  ```
  vsim -sdfmin /top/u1=myasic.sdf
  ```

  Use multiple switches to annotate multiple instances:

  ```
  vsim -sdfmin /top/u1=sdf1 -sdfmin /top/u2=sdf2 top
  ```

- This example searches the libraries *mylib* for *top(only)* and *gatelib* for *cache_set*. If the design units are not found, the search continues to the work library. Specification of the architecture (*only*) is optional.

  ```
  vsim ’mylib.top(only)’ gatelib.cache_set
  ```

- Invoke vsim on *test_counter* and run the simulation until a break event, then quit when it encounters a $finish task.

```
vsim -do "set PrefMain(forceQuit) 1; run -all" work.test_counter
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Echoing of the command line is disabled

```
vsim -stats=time,-cmd,msg
```

- The first -stats option is ignored. The none option disables all modelsim.ini settings and then enables the perf option.

```
vsim -stats=time,cmd,msg -stats=none,perf
```

vsim<info>

The vsim<info> commands return information about the current vsim executable.

Syntax

vsim [Auth | Date | Id | Version | VersionString]

Arguments

- vsimAuth

  Returns the authorization level (PE/SE, VHDL/Verilog/PLUS).
- vsimDate

  Returns the date the executable was built, such as "Apr 10 2000".
- vsimId

  Returns the identifying string, such as "Questa SIM 6.1".
- vsimVersion

  Returns the version as used by the licensing tools, such as "1999.04".
- vsimVersionString

  Returns the full vsim version string.

This same information can be obtained using the -version argument of the vsim command.

vsim_break

Stop (interrupt) the current simulation before it runs to completion. To stop a simulation and then resume it, use this command in conjunction with run -continue.

Syntax

vsim_break

Arguments

None.

Examples

- Interrupt a simulation, then restart it from the point of interruption.

```
vsim_break
run -continue
```

vsource

This command specifies an alternative file to use for the current source file.

This command is used when the current source file has been moved. The alternative source mapping exists for the current simulation only.

Syntax

vsource [<filename>]

Arguments

- <filename>

  (optional) Specifies a relative or full pathname. If filename is omitted, the source file for the current design context is displayed.

Examples

```
vsource design.vhd
vsource /old/design.vhd
```

wave

A number of commands are available to manipulate and report on the Wave window.

The following tables summarize the available options for manipulating cursors, for zooming, and for adjusting the wave display view in the Wave window:

Table 2-17. Wave Window Commands for Cursor

Cursor Commands	Description
wave cursor active	Sets the active cursor to the specified cursor or, if no cursor is specified, reports the active cursor
wave cursor add	Adds a new cursor at specified time and returns the number of the newly added cursor
wave cursor configure	Sets or reports values for the specified cursor
wave cursor delete	Deletes the specified cursor or, if no cursor is specified, the active cursor
wave cursor see	Positions the wave display such that the specified or active cursor appears at the specified percent from the left edge of the display – 0% is the left edge, 100% is the right edge.
wave cursor time	Moves or reports the time of the specified cursor or, if no cursor is specified, the time of the active cursor

Table 2-18. Wave Window Commands for Expanded Time Display

Display view Commands	Description
wave expand mode	Selects the expanded time display mode: Delta Time, Event Time, or off.
wave expand all	Expands simulation time into delta time steps if Delta Time mode is currently selected (WLFCollapseMode = 1) or into event time steps if Event Time mode is currently selected (WLFCollapseMode = 0) over the full range of the simulation from time 0 to the current time.
wave expand cursor	Expands simulation time into delta time steps if Delta Time mode is currently selected (WLFCollapseMode = 1) or into event time steps if Event Time mode is currently selected (WLFCollapseMode = 0) at the simulation time of the active cursor.
wave expand range	Expands simulation time into delta time steps if Delta Time mode is currently selected (WLFCollapseMode = 1) or into event time steps if Event Time mode is currently selected (WLFCollapseMode = 0) over a time range specified by a start time and an end time.

Table 2-18. Wave Window Commands for Expanded Time Display (cont.)

Display view Commands	Description
wave collapse all	Collapses simulation time over the full range of the simulation from time 0 to the current time.
wave collapse cursor	Collapses simulation time at the time of the active cursor.
wave collapse range	Collapses simulation time over a specific simulation time range.

Table 2-19. Wave Window Commands for Controlling Display

Display view Commands	Description
wave interrupt	Immediately stops wave window drawing
wave refresh	Cleans wave display and redraws waves
wave cursor see	Positions the wave display such that the specified or active cursor appears at the specified percent from the left edge of the display – 0% is the left edge, 100% is the right edge.
wave seetime	Positions the wave display such that the specified time appears at the specified percent from the left edge of the display – 0% is the left edge, 100% is the right edge.

Table 2-20. Wave Window Commands for Zooming

Zooming Commands	Description
wave zoom in	Zoom in the wave display by the specified factor. The default factor is 2.0.
wave zoom out	Zoom out the wave display by the specified factor. The default factor is 2.0.
wave zoom full	Zoom the wave display to show the full simulation time.
wave zoom last	Return to last zoom range.
wave zoom range	Sets left and right edge of wave display to the specified start time and end time. If times are not specified, reports left and right edge times.

Syntax

wave cursor active [-window <win>] [<cursor-num>]

wave cursor add [-window <win>] [-time <time>] [-name <name>] [-lock <0 |1>]

wave cursor configure [<cursor-num>] [-window <win>] [<option> [<value>]]

wave cursor delete [-window <win>] [<cursor-num>]

wave cursor see [-window <win>] [-at <percent>] [<cursor-num>]

wave cursor time [-window <win>] [-time <time>] [<cursor-num>]

wave collapse all [-window <win>]

wave collapse cursor [-window <win>] [<cursor-num>]

wave collapse range [-window <win>] <start-time> <end-time>

wave expand all [-window <win>]

wave expand cursor [-window <win>] [<cursor-num>]

wave expand mode [-window <win>] [off | deltas | events]

wave expand range [-window <win>] <start-time> <end-time>

wave interrupt [-window <win>]

wave refresh [-window <win>]

wave seetime [-window <win>] [-at <percent>] -time <time>

wave zoom in [-window <win>] [<factor>]

wave zoom out [-window <win>] [<factor>]

wave zoom full [-window <win>]

wave zoom last [-window <win>]

wave zoom range [-window <win>] [<start-time> <end-time>]

Arguments

- -at <percent>

  (optional) Positions the display such that the time or cursor is the specified <percent> from the left edge of the wave display.

  <percent> — Any non-negative number where the default is 50. 0 is the left edge of the Wave window and 100 is the right edge.

- <cursor-num>

  (optional) Specifies a cursor number. If not specified, the active cursor is used.

- <factor>

  (optional) A number that specifies how much you want to zoom into or out of the wave display. Default value is 2.0.

- -lock <0 |1>

  (optional) Specify the lock state of the cursor.

  0 — (default) Unlocked

  1 — Locked

- -name <name>

  (optional) Specify the name of the cursor.

  <name> — Any string where the default is "Cursor <n>" where <n> is the cursor number.

- off | deltas | events

  (optional) Specifies the expanded time display mode for the Wave window. Default is off.

- <option> [<value>]

  (optional) Specify a value for the designated option. Currently supported options are -name, -time, and -lock. If no option is specified, current value of all options are reported.

- <start-time> <end-time>

  (optional) start-time and end-time are times that specify an expand, collapse, or zoom range. If neither number is specified, the command returns the current range.

- -time <time>

  (optional) Specifies a cursor time.

  <time> — Any positive integer.

- -window <win>

  (optional) All commands default to the active Wave window unless this argument is used to specify a different Wave window.

  <win> — Specifies the name of a Wave window other than the current active window.

Examples

- Either of these commands creates a zoom range with a start time of 20 ns and an end time of 100 ns.

  ```
  wave zoom range 20ns 100ns
  wave zoom range 20 100
  ```

- Return the name of cursor 2:

  ```
  wave cursor configure 2 -name
  ```

- Name cursor 2, "reference cursor" and return that name with:

  ```
  wave cursor configure 2 -name {reference cursor}
  ```

- Return the values of all wave cursor configure options for cursor 2:

  ```
  wave cursor configure 2
  ```

wave create

This command generates a waveform known only to the GUI. You can then modify the waveform interactively or with the wave edit command and use the results to drive simulation.

Refer to "Generating Stimulus with Waveform Editor" for more information.

Some arguments to this command are order-dependent. Please read through the argument descriptions for more information. d

The following table summarizes the available waveform pattern options:

Command	Description
wave create -pattern clock	Generates a clock waveform. Recommended that you specify an initial value, duty cycle, and clock period for the waveform.
wave create -pattern constant	Generates a waveform with a constant value. It is suggested that you specify a value.
wave create -pattern random	Generates a random waveform based upon a seed value. Specify the type (normal or uniform), an initial value, and a seed value. If you don't specify a seed value, Questa uses a default value of 5.
wave create -pattern repeater	Generates a waveform that repeats. Specify an initial value and pattern that repeats. You can also specify how many times the pattern repeats.
wave create -pattern counter	Generates a waveform from a counting pattern. Specify start and end values, repeat, step count, time period, and type (Binary, Gray, Johnson, OneHot, Range, and ZeroHot).
wave create -pattern none	Creates a placeholder for a waveform. Specify an object name.

Syntax

All waveforms

wave create **[-driver {freeze | deposit | driver | expectedoutput}]** [-initialvalue <value>] [-language {vhdl | verilog}] [-portmode {input | output | inout | internal}] [-range <msb lsb>]
[-starttime {<time><unit>}] [-endtime {<time><unit>}] **<object_name>**

Clock patterns

wave create **-pattern clock [-dutycycle <value>] [-period {<time><unit>}] <object_name>**

Constant patterns

wave create **-pattern constant** [-initialvalue <value>] **[-value <value>] <object_name>**

Random patterns

wave create **-pattern random** [-initialvalue <value>] **[-period {<time><unit>}]**
[-random_type {normal | uniform | poisson | exponential}] [-seed <value>]
<object_name>

Repeater patterns

wave create **-pattern repeater** [-initialvalue <value>] **[-period {<time><unit>}]**
[-repeat {forever | never | <n>}] [-sequence {<val1>} <val2> …]

Counter patterns

wave create **-pattern counter** [**-direction {up | down | upthendown | downthenup}]**
[-initialvalue <value>] **[-period {<time><unit>}] [-repeat {forever | never | <n>}]**
[-startvalue <value>] [-endvalue <value>] [-step <value>]
[**-type {binary | gray | johnson | onehot | range | zerohot}] <object_name>**

No pattern

wave create **-pattern none <object_name>**

Arguments

- -pattern clock | constant | random | repeater | counter | none

  (required) Specifies the waveform pattern. Refer to "Accessing the Create Pattern Wizard" for a description of the pattern types.

  clock — Specifies a clock pattern.

  constant — Specifies a constant pattern.

  random — Specifies a random pattern.

  repeater — Specifies a repeating pattern.

  counter — Specifies a counting pattern.

  none — Specifies a blank pattern.

- -direction {up | down | upthendown | downthenup}

  (optional, recommended when specifying -pattern counter) The direction in which the counter will increment or decrement.

  up — (default) Increment only.

  down — Decrement only.

  upthendown — Increment then decrement.

  downthenup — Decrement then increment.

- -driver {freeze | deposit | driver | expectedoutput}

  (optional) Specifies that the signal is a driver of the specified type. Applies to waveforms created with -portmode inout or -portmode internal.

- -dutycycle <value>

  (optional, recommended for -pattern clock) Specifies the duty cycle of the clock. Expressed as a percentage of the period that the clock is high.

  <value> — Any integer from 0 to 100 where the default is 50.

- -endtime {<time><unit>}

  (optional) The simulation time where the waveform will stop. If omitted, the waveform stops at 1000 simulation time units.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -endvalue <value>

  (optional, recommended when specifying -pattern counter) The end value for the counter. This option applies to patterns specifying -type Range only. All other counter patterns start from 0 and go to the maximum value for that particular signal (for example, for a 3-bit signal, the start value will be 000 and the end value will be 111).

  <value> — Value must be appropriate for the type of waveform you are creating.

- -initialvalue <value>

  (optional) The initial value for the waveform. Not applicable to counter patterns.

  <value> — Value must be appropriate for the type of waveform you are creating.

- -language {vhdl | verilog}

  (optional) Controls which language is used for the created wave.

  vhdl — (default) Specifies the VHDL language.

  verilog — Specifies the Verilog language.

- -period {<time><unit>}

  (optional, recommended for all patterns except -constant) Specifies the period of the signal.

  <time> — Specified as an integer or decimal number. Current simulation units are the default unless specifying <unit>.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -portmode {input | output | inout | <u>internal</u>}

  (optional) The port type for the waveform. Useful for creating signals prior to loading a design.

  in — Ports of type IN. You can also specify “input” as an alias for in.

out — Ports of type OUT. You can also specify “output” as an alias for out.

inout — Ports of type INOUT.

internal — (default) Ports of type INTERNAL.

- -random_type {normal | uniform | poisson | exponential}

  (optional, recommended when specifying -pattern random) Specifies the type of algorithm used to generate a random waveform pattern.

  normal — Normal or Gaussian distribution of waveform events.

  uniform — (default) Uniform distribution of waveform events.

  poisson — Poisson distribution of waveform events.

  exponential — Exponential distribution of waveform events.

- -range <msb lsb>

  (optional) Identifies bit significance in a counter pattern.

  msb lsb — Most significant bit and least significant bit. Both must be specified.

- -repeat {forever | never | <n>}

  (optional, recommended when specifying -pattern repeater or -pattern counter) Controls duration of pattern repetition.

  forever — Repeat the pattern for as long as the simulation runs.

  never — Never repeat the pattern during simulation.

  <n> — Repeat the pattern <n> number of times where <n> is any positive integer.

- -seed <value>

  (optional, recommended when specifying -pattern random) Specifies a seed value for a randomly generated waveform.

  <value> — Any non-negative integer where the default is 5.

- -sequence {<val1>} <val2> …

  (optional, recommended when specifying pattern -repeater) The set of values that you want repeated.

  <val1> — Value must be appropriate for the type of waveform you are creating. Multiple values are entered as a space separated list and enclosed in curly braces ({}).

- -starttime {<time><unit>}

  (optional) The simulation time at which the waveform should start. If omitted, the waveform starts at 0 simulation time units.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -startvalue <value>

  (required when specifying -pattern counter) The initial value of the counter. This option applies to patterns specifying -type Range only. All other counter patterns start from 0 and go to the maximum value for that particular signal (e.g., for a 3-bit signal, the start value will be 000 and the end value will be 111).

  <value> — Value must be appropriate for the type of waveform you are creating.

- -step <value>

  (optional, recommended when specifying -pattern counter) The step by which the counter is incremented/decremented.

  <value> — Value must be appropriate for the type of waveform you are creating.

- -type {binary | gray | johnson | onehot | range | zerohot}

  (optional) Specifies a counter format.

  binary — Specifies a binary counter.

  gray — Specifies a binary counter where two successive values differ in only one bit. Also known as a reflected binary counter.

  johnson — Specifies a twisted ring or Johnson counter.

  onehot — Specifies a shift counter where only one bit at a time is set to “on” (1).

  range — (default) Specifies a binary counter where the values range between -startvalue and -endvalue

  zerohot — Specifies a shift counter where only one bit at a time is set to “off” (0).

- -value <value>

  (optional, recommended when specifying -pattern constant) Specifies a value for the constant pattern.

  <value> — Value must be appropriate for the type of waveform you are creating.

- <object_name>

  (required) User specified name for the waveform. Must be the final argument.

Examples

- Create a clock signal with the following default values:

  wave create -pattern clock -period 100 -dutycycle 50 -starttime 0 -endtime 1000 -initialvalue 0 /counter/clk

- Create a constant 8-bit signal vector from 0 to 1000 ns with a value of 1111 and a drive type of freeze.

  wave create -driver freeze -pattern constant -value 1111 -range 7 0 -starttime 0ns -endtime 1000ns sim:/andm/v_cont2

Related Topics

- wave edit
- wave modify
- "Generating Stimulus with Waveform Editor"
- "Accessing the Create Pattern Wizard"

wave edit

This command modifies waveforms created with the wave create command.

Some arguments to this command are order-dependent. Please read through the argument descriptions for more information. d

The following table summarizes the available editing options:

Command	Description
wave edit cut	Cut part of a waveform to the clipboard
wave edit copy	Copy part of a waveform to the clipboard
wave edit paste	Paste the waveform from the clipboard
wave edit invert	Vertically flip part of a waveform
wave edit mirror	Mirror part of a waveform
wave edit insert_pulse	Insert a new edge on a waveform; doesn't affect waveform duration
wave edit delete	Delete an edge from a waveform; doesn't affect waveform duration
wave edit stretch	Move an edge by stretching the waveform
wave edit move	Move an edge without moving other edges
wave edit change_value	Change the value of part of a waveform
wave edit extend	Extend all waves
wave edit driveType	Change the driver type
wave edit undo	Undo an edit
wave edit redo	Redo a previously undone edit

Syntax

wave edit **{cut | copy | paste | invert | mirror} -end {<time><unit>} -start {<time><unit>} <object_name>**

wave edit insert_pulse [-duration {<time><unit>}] **-start {<time><unit>} <object_name>**

wave edit delete **-time {<time><unit>} <object_name>**

wave edit stretch | move **{-backward {<time><unit>}** | **-forward {<time><unit>}} -time {<time><unit>} <object_name>**

wave edit change_value **-end {<time><unit>} -start {<time><unit>} <value> <object_name>**

wave edit extend **-extend to | by -time {<time><unit>}**

wave edit driveType **-driver freeze | deposit | driver | expectedoutput -end {<time><unit>} -start {<time><unit>}**

wave edit undo **<number>**

wave edit redo **<number>**

Arguments

- -backward {<time><unit>}

  (required if -forward <time> isn't specified) The amount to stretch or move the edge backwards in simulation time.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- cut | copy | paste | invert | mirror

  (required) Specifies the type of edit to perform.

  cut — Deletes the specified portion of the waveform.

  copy — Saves a copy of the specified portion of the waveform.

  paste — Inserts the contents of the clipboard into the specified portion of the waveform.

  invert — Flips the specified portion of the waveform vertically.

  mirror — Flips the specified portion of the waveform horizontally.

- -driver freeze | deposit | driver | expectedoutput

  (required) Specifies the type of driver to which you want the specified section of the waveform changed. Applies to signals of type inout or internal.

- -duration {<time><unit>}

  (optional) The length of the pulse.

  <time> — Specified as an integer or decimal number where the default is 10 time units.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -end {<time><unit>}

  (required unless specifying paste) The end of the section of waveform to perform the editing operation upon, denoted by a simulation time.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -extend to | by

  (required) Specifies the format for extending waves.

  to — Extends the wave to the time specified by -time <time>.

  by — Extends the wave by the amount of time specified by -time <time>.

- -forward {<time><unit>}

  (required if -backward <time> is not specified) The amount to stretch or move the edge forwards in simulation time.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- <number>

  (optional) The number of editing operations to undo or redo. If omitted, only one editing operation is undone or redone.

- <object_name>

  (required) The pathname of the waveform to edit. Must be specified as the last argument to wave edit.

- -start {<time><unit>}

  (required) The beginning of the section of waveform to perform the editing operation upon, denoted by a simulation time.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -time {<time><unit>}

  (required) The amount of time to extend or stretch waves.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- <value>

  (required) The new value. Must match the type of the <object_name>.

Related Topics

- wave create
- "Generating Stimulus with Waveform Editor"

wave export

This command creates a stimulus file from waveforms created with the wave create command.

Syntax

wave export **-designunit <name>** -starttime {<time><unit>} **-endtime {<time><unit>} -file <filename>** **{-format force | vcd | vhdl | verilog}**

Arguments

- -designunit <name>

  (required) Specifies a design unit for which you want to export created waves. If omitted, the command exports waves from the active design unit.

  <name> — Specifies a design unit in the simulation.

- -endtime {<time><unit>}

  (required) The simulation time at which you want to stop exporting.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -file <filename>

  (required) The filename for the saved export file.

  <name> — Any user specified string.

- -format force | vcd | vhdl | verilog

  (required) The format of the saved stimulus file. The format options include:

  force — A Tcl script that recreates the waveforms. The file should be sourced when reloading the simulation.

  vcd — An extended VCD file. Load using the -vcdstim argument to vsim.

  vhdl — A VHDL test bench. Compile and load the file as your top-level design unit.

  verilog — A Verilog test bench. Compile and load the file as your top-level design unit.

- -starttime {<time><unit>}

  (required) The simulation time at which you want to start exporting.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

Related Topics

- wave create
- wave import
- "Generating Stimulus with Waveform Editor"

wave import

This command imports an extended VCD file that was created with the wave export command. It cannot read extended VCD file created by software other than Questa SIM. Use this command to apply a VCD file as stimulus to the current simulation.

Syntax

wave import <VCD_file>

Arguments

- <VCD_file>

  (required) The name of the extended VCD file to import.

Related Topics

- wave create
- wave export
- "Generating Stimulus with Waveform Editor"

wave modify

This command modifies waveform parameters set by a previous wave create command.

Some arguments to this command are order-dependent. Please read through the argument descriptions for more information.

The following table summarizes the available wave modification options:

Command	Description
wave modify -pattern clock	Generates a clock waveform. Specify an initial value, duty cycle, and clock period for the waveform.
wave modify -pattern constant	Generates a waveform with a constant value. Specify a value.
wave modify -pattern counter	Generates a waveform from a counting pattern. Specify start and end values, repeat, step count, time period, and type (Binary, Gray, Johnson, OneHot, Range, and ZeroHot).
wave modify -pattern random	Generates a random waveform based upon a seed value. Specify the type (normal or uniform), an initial value, and a seed value. If you don't specify a seed value, Questa uses a default value of 5.
wave modify -pattern repeater	Generates a waveform that repeats. Specify an initial value and pattern that repeats. You can also specify how many times the pattern repeats.
wave modify -pattern none	Creates a placeholder for a waveform. Specify an object name.

Syntax

All waveforms

wave modify [-driver freeze | deposit | driver | expectedoutput] [-endtime {<time><unit>}] [-initialvalue <value>] [-portmode {input | output | inout | internal}] [-range <msb lsb>] [-starttime {<time><unit>}] **<wave_name>**

Clock patterns only

wave modify -pattern clock **-period <value> -dutycycle <value> <wave_name>**

Constant patterns only

wave modify -pattern constant [-driver freeze | deposit | driver | expectedoutput] [-language {vhdl | verilog}] [**-value <value>**] **<wave_name>**

Counter patterns only

wave modify -pattern counter **-period <value> -repeat forever | <n> | never -startvalue <value> -step <value>** [-direction {up | down | upthendown | downthenup}]

[-endvalue <value>] [-type {binary | gray | johnson | onehot | range | zerohot}] **<wave_name>**

Random patterns only

wave modify -pattern random **-period <value>** **-random_type exponential | normal | poisson | uniform** [-seed <value>] **<wave_name>**

Repeater patterns only

wave modify -pattern repeater **-period <value> -repeat forever | <n> | never** **-sequence {val1 val2 val3 …} <wave_name>**

No pattern

wave create **-pattern none <wave_name>**

Arguments

- -direction {up | down | upthendown | downthenup}

  (optional, recommended when specifying -pattern counter) The direction in which the counter will increment or decrement.

  up — (default) Increment only.

  down — Decrement only.

  upthendown — Increment then decrement.

  downthenup — Decrement then increment.

- -driver freeze | deposit | driver | expectedoutput

  (optional) Specifies that the signal is a driver of the specified type. Applies to signals of type inout or internal.

- -dutycycle <value>

  (required) The duty cycle of the clock, expressed as a percentage of the period that the clock is high.

  <value> — Any integer from 0 to 100 where the default is 50.

- -endtime {<time><unit>}

  (optional) The simulation time that the waveform should stop. If omitted, the waveform stops at 1000 simulation time units.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -endvalue <value>

  (optional) The ending value of the counter. This option applies to Range counter patterns only. All other counter patterns start from 0 and go to the max value for that particular signal (for example, for a 3-bit signal, the start value will be 000 and end value will be 111).

  <value> — Any positive integer.

- -initialvalue <value>

  (optional) The initial value for the waveform. Value must be appropriate for the type of waveform you are creating. Not applicable to counter patterns.

  <value> — Any positive integer.

- -language {vhdl | verilog}

  (optional) Controls which language is used for modifying the wave.

  vhdl — (default) Specifies the VHDL language.

  verilog — Specifies the Verilog language.

- -period <value>

  (required) The period of the signal.

- -portmode {input | output | inout | internal}

  (optional) The port type for the waveform.

  in — Ports of type IN. You can also specify “input” as an alias for in.

  out — Ports of type OUT. You can also specify “output” as an alias for out.

  inout — Ports of type INOUT.

  internal — (default) Ports of type INTERNAL.

- -random_type exponential | normal | poisson | uniform

  (required) Specifies a random pattern to generate.

  exponential — Exponential distribution of waveform events.

  normal — Normal or Gaussian distribution of waveform events.

  poisson — Poisson distribution of waveform events.

  uniform — (default) Uniform distribution of waveform events.

- -range <msb lsb>

  (optional) Identifies bit significance in a counter pattern.

  msb lsb — Most significant bit and least significant bit. Both must be specified.

- -repeat forever | <n> | never

  (required) Controls duration of pattern repetition.

  forever — Repeat the pattern for as long as the simulation runs.

  <n> — Repeat the pattern <n> number of times where <n> is any positive integer.

never — Never repeat the pattern during simulation.

- -seed <value>

  (optional) Specifies a seed value for a randomly generated waveform.

  <value> — Any non-negative integer where the default is 5.

- -sequence {val1 val2 val3 …}

  (required) The set of values that you want repeated.

  <val1> — Value must be appropriate for the type of waveform you are creating. Multiple values are entered as a space separated list and enclosed in curly braces ({}).

- -starttime {<time><unit>}

  (optional) The simulation time that the waveform should start. If omitted, the waveform starts at 0 simulation time units.

  <time> — Specified as an integer or decimal number.

  <unit> — (optional) A suffix specifying a unit of time where the default is to specify the current simulation resolution by omitting <unit>. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}).

- -startvalue <value>

  (required when specifying -pattern counter) The initial value of the counter. This option applies to patterns specifying -type Range only. All other counter patterns start from 0 and go to the maximum value for that particular signal (e.g., for a 3-bit signal, the start value will be 000 and the end value will be 111).

  <value> — Value must be appropriate for the type of waveform you are creating.

- -step <value>

  (required) The step by which the counter is incremented/decremented.

  <value> — Value must be appropriate for the type of waveform you are creating.

- -type {binary | gray | johnson | onehot | range | zerohot}

  (optional) Specifies a counter format.

  binary — Specifies a binary counter.

  gray — Specifies a binary counter where two successive values differ in only one bit. Also known as a reflected binary counter.

  johnson — Specifies a twisted ring or Johnson counter.

  onehot — Specifies a shift counter where only one bit at a time is set to “on” (1).

  range — (default) Specifies a binary counter where the values range between -startvalue and -endvalue

  zerohot — Specifies a shift counter where only one bit at a time is set to “off” (0).

- -value <value>

  (optional, recommended when specifying -pattern constant) Specifies a value for the constant pattern.

  <value> — Value must be appropriate for the type of waveform you are creating.

- <wave_name>

  (required) The name of an existing waveform created with the wave create command.

Related Topics

- wave create
- "Generating Stimulus with Waveform Editor"
- "Accessing the Create Pattern Wizard".

wave sort

This command sorts signals in the Wave window by name or full path name.

Syntax

wave sort **{ascending | descending | fa | fd}** [-win <window_name>]

Arguments

- ascending | descending | fa | fd

  (required) Sort signals in one of the following orders.

  ascending — Sort in ascending order by signal name.

  descending — Sort in descending order by signal name.

  fa — Sort in ascending order by the full path name.

  fd — Sort in descending order by full path name.

- -win <window_name>

  (optional) Specifies an instance of the Wave window that is not the default. Otherwise, the default Wave window is used. Use the view command to change the default window.

  <window_name> — Window name other than the default Wave window.

Examples

```
wave sort ascending
```

when

This command instructs Questa SIM to perform actions when the specified conditions are met.

For example, you can use the command to break on a signal value or at a specific simulator time. Use the nowhen command to deactivate when commands.

Note

When running in full optimization mode, breakpoints cannot be set. Run the design in non-optimized mode (or set +acc arguments) to enable you to set breakpoints in the design. Refer to Preserving Object Visibility for Debugging Purposes.

Syntax

when [[-fast] [-id <id#>] [-label <label>] **{<when_condition_expression>} {<command>}**]

Description

The when command uses a when_condition_expression to determine whether or not to perform the action. Conditions can include VHDL signals and Verilog nets and registers. The when_condition_expression uses a simple restricted language (that is not related to Tcl), which permits only four operators and operands that may be either HDL object names, signame'event, or constants. Questa SIM evaluates the condition every time any object in the condition changes, hence the restrictions.

Here are some additional points to keep in mind about the when command:

- The when command creates the equivalent of a VHDL process or a Verilog always block. It does not work like a looping construct you might find in other languages such as C.
- Virtual signals, functions, regions, types, and so forth, cannot be used in the when command. Neither can simulator state variables other than $now.
- With no arguments, when will list the currently active when statements and their labels (explicit or implicit).

Embedded Commands Allowed with the -fast Argument

You can use any Tcl command as a <command>, along with any of the following vsim commands:

- bp, bd
- change
- disablebp, enablebp
- echo
- examine
- force, noforce

- log, nolog
- stop
- when, nowhen

Embedded Commands Not Allowed with the -fast Argument

- Any do commands
- Any Tk commands or widgets
- References to U/I state variables or tcl variables
- Virtual signals, functions, or types

Using Global Tcl Variables with the -fast Argument

Embedded commands that use global Tcl variables for passing a state between the when command and the user interface need to declare the state using the Tcl uivar command. For example, the variable i below is visible in the GUI. From the command prompt, you can display it (by entering echo $i) or modify it (for example, by entering set i 25).

```
set i 10
when -fast {clk == '0'} {
   uivar i
   set i [expr {$i - 1}]
   if {$i <= 0} {
      force reset 1 0, 0 250
   }
}
when -fast {reset == '0'} {
   uivar i
   set i 10
}
```

Additional Restrictions on the -fast Argument

Accessing channels (such as files, pipes, sockets) that were opened outside of the embedded command will not work. For example:

```
set fp [open mylog.txt w]
when -fast {bus} {
   puts $fp "bus change: [examine bus]"
}
```

The channel that $fp refers to is not available in the simulator, only in the user interface. Even using the uivar command does not work here because the value of $fp has no meaning in the context of the -fast argument.

The following method of rewriting this example opens the channel, writes to it, then closes it within the when command:

```
when -fast {bus} {
   set fp [open mylog.txt a]
   puts $fp "bus change: [examine bus]"
   close $fp
}
```

The following example is a little more sophisticated method of doing the same thing:

```
when -fast {$now == 0ns} {
   set fp [open mylog.txt w]
}
when -fast {bus} {
   puts $fp "bus change: [examine bus]"
}
when -fast {$now == 1000ns} {
   close $fp
}
```

The general principle is that any embedded command done using the -fast argument is global to all other commands used with the -fast argument. Here, {$now == 0ns} is a way to define Tcl processes that the -fast commands can use. These processes have the same restrictions that when bodies have, but the advantage is again speed as a proc will tend to execute faster than code in the when body itself.

It is recommended not to use virtual signals and expressions.

Arguments

- -fast

  (optional) Causes the embedded <command> to execute within the simulation kernel, which provides faster execution and reduces impact on simulation runtime performance. Limitations on using the -fast argument are described above (in "Embedded Commands Not Allowed with the -fast Argument"). Disallowed commands still work, but they slow down the simulation.

- -label <label>

  (optional) Used to identify individual when commands.

  <label> — Associates a name or label with the specified when command. Adds a level of identification to the when command. The label may contain special characters. Quotation marks (" ") or braces ({ }) are required only if <label> contains spaces or special characters.

- -id <id#>

  (optional) Attempts to assign this id number to the when command.

  <id#> — Any positive integer that is not already assigned. If the id number you specify is already used, Questa SIM will return an error.

Note

Id numbers for when commands are assigned from the same pool as those used for the bp command. So even if you have not specified a given id number for a when command, that number may still be used for a breakpoint.

- {<when_condition_expression>}

  (required if a command is specified) Specifies the conditions to be met for the specified <command> to be executed. The condition is evaluated in the simulator kernel and can be an object name, in which case the curly braces can be omitted. The command will be executed when the object changes value. The condition can be an expression with these operators:

Name	Operator
equals	==, =
not equal	!=, /=
greater than	>
less than	<
greater than or equal	>=
less than or equal	<=
AND	&&, AND
OR	||, OR

  The operands may be object names, signame'event, or constants. Subexpressions in parentheses are permitted. The command will be executed when the expression is evaluated as TRUE or 1.

  The formal BNF syntax is:

```
condition ::= Name | { expression }

expression ::= expression AND relation
             | expression OR relation
             | relation
relation ::= Name = Literal
           | Name /= Literal
           | Name ' EVENT
           | ( expression )
Literal ::= '<char>' | "<bitstring>" | <bitstring>
```

  The "=" operator can occur only between a Name and a Literal. This means that you cannot compare the value of two signals, i.e., Name = Name is not possible.

  Tcl variables can be used in the condition expression but you must replace the curly braces ({}) with double quotes ("). This works like a macro substitution where the Tcl variables are evaluated once and the result is then evaluated as the when condition. Condition expressions are evaluated in the vsim kernel, which knows nothing about Tcl variables.

That's why the condition expression must be evaluated in the GUI before it is sent to the vsim kernel. See below for an example of using a Tcl variable.

The ">", "<", ">=", and "<=" operators are the standard ones for vector types, not the overloaded operators in the std_logic_1164 package. This may cause unexpected results when comparing objects that contain values other than 1 and 0. Questa SIM does a lexical comparison (position number) for values other than 1 and 0. For example:

```
0000 < 1111 ## This evaluates to true
H000 < 1111 ## This evaluates to false
001X >= 0010 ## This also evaluates to false
```

- {<command>}

  (required if a when expression is specified) The command(s) for this argument are evaluated by the Tcl interpreter within the Questa SIM GUI. Any Questa SIM or Tcl command or series of commands are valid with one exception—the run command cannot be used with the when command. The command sequence usually contains a stop command that sets a flag to break the simulation run after the command sequence is completed. Multiple-line commands can be used.

Note

If you want to stop the simulation using a when command, you must use a stop command within your when statement. DO NOT use an exit command or a quit command. The stop command acts like a breakpoint at the time it is evaluated.

Examples

- The when command below instructs the simulator to display the value of object *c* in binary format when there is a clock event, the clock is 1, and the value of *b* is 01100111. Finally, the command tells Questa SIM to stop.

```
when -label when1 {clk'event and clk='1' and b = "01100111"} {
   echo "Signal c is [exa -bin c]"
   stop
}
```

- The when command below echoes the simulator time when slice [3:1] of wire [15:0] count matches the hexadecimal value 7, and simulation time is between 70 and 111 nanoseconds.

```
when {$now > 70ns and count(3:1) == 3'h7 && $now < 111ns} {
   echo "count(3:1) matched 3'h7 at time " $now
}
```

- The commands below show an example of using a Tcl variable within a when command. Note that the curly braces ({}) have been replaced with double quotes ("").

```
set clkb_path /tb/ps/dprb_0/udprb/ucar_reg/uint_ram/clkb;
when -label when1 "$clkb_path'event and  $clkb_path ='1'" {
     echo "Detected Clk edge at path $clkb_path"
}
```

- The when command below is labeled *a* and will cause Questa SIM to echo the message “b changed” whenever the value of the object *b* changes.

```
when -label a b {echo "b changed"}
```

- The multi-line when command below does not use a label and has two conditions. When the conditions are met, Questa SIM runs an echo command and a stop command.

```
when {b = 1
    and c /= 0 } {
    echo "b is 1 and c is not 0"
    stop
}
```

- In the example below, for the declaration "wire [15:0] a;", the when command will activate when the selected bits match a 7:

```
when {a(3:1) = 3'h7} {echo "matched at time " $now}
```

- If you encounter a vectored net caused by optimizing with vopt, use the 'event qualifier to prevent the command from falsely evaluating when unrelated bits of ‘a’ change:

```
when {a(3:1) = 3'h7 and a(3:1)'event} {echo "matched at time " $now}
```

- In the example below, we want to sample the values of the address and data bus on the first falling edge of *clk* after *sstrb* has gone high.

```
# ::strobe is our state variable
set ::strobe Zero
# This signal breakpoint only fires when sstrb changes to a '1'
when -label checkStrobe {/top/sstrb == '1'} {
   # Our state Zero condition has been met, move to state One
   set ::strobe One
   }
# This signal breakpoint fires each time clk goes to '0'
when {/top/clk == '0'} {
   if {$::strobe eq "One"} {
      # Our state One condition has been met
      # Sample the busses
      echo Sample paddr=[examine -hex /top/paddr] :: sdata=[examine
-hex
      /top/sdata]
      # reset our state variable until next rising edge of sstrb
(back to
      state Zero)
      set ::strobe Zero
   }
}
```

Ending the simulation with the stop command

Batch mode simulations are often structured as "run until condition X is true," rather than "run for X time" simulations. The multi-line when command (shown below) sets a done condition, and Questa SIM runs an echo command and a stop command when the condition is reached.

The simulation will not stop (even if a quit -f command is used) unless you enter a stop command. To exit the simulation and quit Questa SIM, use an approach like the following:

```
onbreak {resume}
when {/done_condition == '1'} {
     echo "End condition reached"
     if [batch_mode] {
          set DoneConditionReached 1
          stop
     }
}
run 1000 us
if {$DoneConditionReached == 1} {
     quit -f
}
```

This example stops 100ns after a signal transition:

```
when {a = 1} {
  # If the 100ns delay is already set then let it go.
  if {[when -label a_100] == ""} {
    when -label a_100 { $now = 100 } {
      # delete this breakpoint then stop
      nowhen a_100
      stop
    }
  }
}
```

Time-based breakpoints

You can build time-based breakpoints into a when statement with the following syntax.

For absolute time (indicated by @) use:

```
when {$now = @1750 ns} {stop}
```

You can also use:

```
when {errorFlag = '1' OR $now = 2 ms} {stop}
```

This example adds 2 ms to the simulation time at which the when statement is first evaluated, then stops. The white space between the value and time unit is required for the time unit to be understood by the simulator.

You can also use variables, as shown in the following example:

```
set time 1000
when "\$now = $time" {stop}
```

The quotes instruct Tcl to expand the variables before calling the command. So, the when command sees:

```
when "$now = 1000" stop
```

Note that "$now" has the '$' escaped. This prevents Tcl from expanding the variable, because if it did, you would get:

```
when "0 = 1000" stop
```

Related Topics

- bp
- disablebp
- enablebp
- nowhen

where

This command displays information about the system environment. It is useful for debugging problems where Questa SIM cannot find the required libraries or support files.

The command displays two results on consecutive lines:

- current directory

  This is the current directory that Questa SIM was invoked from, or that was specified on the Questa SIM command line.

- current project file

  This is the *.mpf* file Questa SIM is using. All library mappings are taken from here when a project is open. If the design is not loaded through a project, this line displays the *modelsim.ini* file in the current directory.

Syntax

where

Arguments

None.

Examples

- Design is loaded through a project:

  VSIM> where

  Returns:

  ```
  #  Current directory is: D:\Client
  #  Project is: D:/Client/monproj.mpf
  ```

- Design is loaded with no project (indicates the modelsim.ini file is under the `mydesign` directory):

  VSIM> where

  Returns:

  ```
  #  Current directory is: C:\Client\testcase\mydesign
  #  Project is: modelsim.ini
  ```

wlf2log

This command translates a Questa SIM WLF file (*vsim.wlf*) to a QuickSim II logfile. It reads the *vsim.wlf* WLF file generated by the add list, add wave, or log commands in the simulator and converts it to the QuickSim II logfile format.

Note

This command should be invoked only after you have stopped the simulation using quit -sim or dataset close sim.

Syntax

wlf2log **<wlffile>** [-bits] [-fullname] [-help] [-inout] [-input] [-internal] [-l <instance_path>] [-lower] [-o <outfile>] [-output] [-quiet]

Arguments

- <wlffile>

  (required) Specifies the Questa SIM WLF file that you are converting.

- -bits

  (optional) Forces vector nets to be split into 1-bit wide nets in the log file.

- -fullname

  (optional) Shows the full hierarchical pathname when displaying signal names.

- -help

  (optional) Displays a list of command options with a brief description for each.

- -inout

  (optional) Lists only the inout ports. This may be combined with the -input, -output, or -internal switches.

- -input

  (optional) Lists only the input ports. This may be combined with the -output, -inout, or -internal switches.

- -internal

  (optional) Lists only the internal signals. This may be combined with the -input, -output, or -inout switches.

- -l <instance_path>

  (optional) Lists the signals at or below an HDL instance path within the design hierarchy.

  <instance_path> — Specifies an HDL instance path.

- -lower

  (optional) Shows all logged signals in the hierarchy. When invoked without the -lower switch, only the top-level signals are displayed.

- -o <outfile>

  (optional) Directs the output to be written to a file where the default destination for the logfile is standard out.

  <outfile> — A user specified filename.

- -output

  (optional) Lists only the output ports. This may be combined with the -input, -inout, or -internal switches.

- -quiet

  (optional) Disables error message reporting.

Additional information for QuickSim II users

In some cases your original QuickHDL/Questa SIM simulation results (in your *vsim.wlf* file) may contain signal values that do not directly correspond to qsim_12state values. The resulting QuickSim II logfile generated by wlf2log may contain state values that are surrounded by single quotes, e.g. '0' and '1'. To make this logfile compatible with QuickSim models (that expect qsim_12state) you need to use a QuickSim II function named $convert_wdb().

This function was created to convert logfiles resulting from VHDL simulation that used std_logic and std_ulogic since these data types do not correlate to QuickSim's 12 simulation states. Other VHDL data types such as qsim_state or bit (2 state) do not require conversion as they are directly compatible with qsim_12state QuickSim II Waveform Databases (WDB).

The following procedure can be used to convert a wlf2log-generated logfile into a compatible QuickSim WDB. The procedure below shows how to convert the logfile while loaded into memory in QuickSim II.

1. Load the logfile (the output from wlf2log) into a WDB other than "forces". "Forces" is the default WDB, so you need to choose a unique name for the WDB when loading the logfile (for example, "fred").

2. Enter the following at the command prompt from within QuickSim:

   $convert_wdb("fred",0)

   The first argument, which is "fred," is the name of the new WDB to be created. The second argument, which is 0, specifies the type of conversion. At this time only one type of conversion is supported. The value 0 specifies to convert std_logic or std_ulogic into qsim_12state.

3. Do a connect_wdb (either through the pulldown menus, the "Connect WDB" palette icon under "Stimulus", or a function invocation). You specify the name of the WDB that you originally loaded the logfile into (in this case, "fred").

At this point you can issue the "run" command and the stimulus in the converted logfile will be applied. Before exiting the simulation you should save the new WDB ("fred") as a WDB or logfile so that it can be loaded again in the future. The new WDB or logfile will contain the correct qsim_12state values eliminating the need to re-use $convert_wdb().

wlf2vcd

This command translates a Questa SIM WLF file to a standard VCD file. Complex data types that are unsupported in the VCD standard (records, memories, etc.) are not converted.

Note

This command should be invoked only after you have stopped the simulation using quit -sim or dataset close sim.

Syntax

wlf2vcd **<wlffile>** [-help] [-o <outfile>] [-quiet]

Arguments

- <wlffile>

  (required) Specifies the Questa SIM WLF file that you are converting.

- -help

  (optional) Displays a list of command options with a brief description for each.

- -o <outfile>

  (optional) Specifies a filename for the output where the default destination for the VCD output is stdout.

  <outfile> — A user specified filename.

- -quiet

  (optional) Disables warning messages that are produced when an unsupported type (for example, records) is encountered in the WLF file.

wlfman

This command allows you to get information about and manipulate saved WLF files.

The command performs different functions depending on which mode you use:

- wlfman info returns file information, resolution, versions, and so forth about the source WLF file.
- wlfman items generates a list of HDL objects (i.e., signals) and/or transaction streams from the source WLF file and outputs it to stdout. When redirected to a file, the output is called an object_list_file, and it can be read in by wlfman filter. Comments start with a '#' and continue to the end of the line. Wildcards are legal in the leaf portion of the name. Here is an example:

  ```
  /top/foo     # signal foo
  /top/u1/*    # all signals under u1
  /top/u1      # same as line above
  -r /top/u2   # recursively, all signals under u2
  /top/stream1 # transaction stream stream1
  ```

  object_list_files — The object_list_file is a list of objects and/or transaction streams, one per line. Transaction object lists can include a stream name, stream array name, or sub stream, all with a full path. Transaction objects recorded in the object_list_file:

  Full path of a stream array — Logs and records all of the individual streams in the specified stream array, the sub-streams of each stream, and the phase sub-streams recursively for each sub-stream including the attributes of the transactions present in the sub-streams and phase sub-streams. For example:

  ```
  /top/<stream_array> # stream array full path (ex: /top/stream_array)
  ```

  Full path of a stream object — Logs and records all of the sub-streams and recursively records all phase sub-streams for each specified sub-stream including the attributes of the transactions present in the sub-streams and phase sub-streams. For example:

  ```
  /top/<stream_object> # individual stream full path (ex: /top/stream)
  ```

  Sub-streams — Logs and records all of the attributes of the specified sub-stream. Other sub-streams of the main stream are not logged. Phase sub-streams cannot be individually logged. For example:

  ```
  /top/<stream_object> # individual stream full path (ex: /top/stream)
  /top/<stream_object>.<sub-stream> # sub-stream full path
     (ex: /top/stream.s0)
  ```

Note

You can produce these files from scratch but be careful with syntax. It is recommended that you use wlfman items as it always creates a legal object_list_file.

- wlfman filter reads in a WLF file and, optionally, an object_list_file, and writes a new WLF file containing filtered information from those sources. You determine the filtered information with the arguments you specify.
- wlfman monitor returns the current state of a WLF file to the transcript. Each time the state is monitored, a line of information is output. The state of the WLF file can be monitored at regular intervals, indicating the changes over time. Here is an example:

```
wlfman monitor visim.wlf
File    Sim
State   Time
closed  14000
```

- wlfman profile generates a report of the estimated percentage of file space that each signal is taking in the specified WLF file. This command can identify signals that account for a large percentage of the WLF file size (such as a logged memory that uses a zero-delay integer loop to initialize the memory). You may be able to drastically reduce WLF file size by not logging those signals.

  When the WLF file contains transaction streams and/or stream arrays, wlfman profile generates an additional report of estimated file space used for each transaction as a percentage of total file size. You may be able to drastically reduce WLF file size by not logging some transactions or streams.

  The stated size of a transaction is equal to the size of the transaction without any user attributes plus the sum of the sizes of every attribute in that transaction. Also, an assumption is made that every transaction will have its attributes recorded if one of the transactions in a sub-stream has that attribute. If the object is a stream array, the sum of the sizes of all the streams is presented in which case the sizes of the individual elements are not presented.
- wlfman merge combines two WLF files with different signals or transaction objects into one WLF file. It *does not* combine wlf files containing the same signals at different runtime ranges (for example, mixedhdl_0ns_100ns.wlf & mixedhdl_100ns_200ns.wlf). When merging two WLF files containing the same transaction streams, the first stream's data is recorded, the second stream ignored, and a warning issued that a horizontal merge is not supported.
- wlfman optimize copies the data from the WLF file to the output WLF file, adding or replacing the indexing and optimization information.

The different modes are intended to be used together. For example, you might run wlfman profile and identify a signal or transaction stream that accounts for 50% of the WLF file size. If you don't actually need that object, you can then run wlfman filter to remove it from the WLF file.

Syntax

wlfman info **<source_wlffile>** [-v]

wlfman items **<source_wlffile>** [-n] [-v]

wlfman filter **-o <out_wlffile> <source_wlffile>** [-begin <time>] [-end <time>]
[-collapsedelta | -collapsetime | -nocollapse] [-compress | -nocompress]
[-f <object_list_file>] [-index | -noindex] [-r <object>] [-nowarn <number>] [-opt | -noopt]
[-s <symbol>] [-t <resolution>]

wlfman profile **<source_wlffile>** [-rank] [-top <number>]

wlfman merge **-o <out_wlffile>** [**<wlffile1> <wlffile2> ...**] [-compress | -nocompress]
[-index | -noindex] [-opt | -noopt]

wlfman monitor [-f | -i <intervalTime> | -p <endTime>] [-q | -v] **<source_wlffile>**

wlfman optimize **-o <out_wlffile> <source_wlffile>** [-compress | -nocompress]
[-index | -noindex] [-opt | -noopt]

Arguments

- -o <out_wlffile>

  (required) Specifies the name of the output WLF file. The output WLF file will contain all objects specified by the preceding arguments. Output WLF files are always written in the latest WLF version regardless of the source WLF file version.

- <source_wlffile>

  (required) Specifies the WLF file from which you want information.

- <wlffile1> <wlffile2> ...

  (required) Specifies the WLF files whose objects you want to copy into one WLF file. Specified as a space separated list.

- -begin <time>

  (optional) Specifies the simulation time at which to start reading information from the source WLF file where the default is to include the entire length of time recorded in <source_wlffile>. If a transaction on a particular stream started prior to the time specified, then that transaction is ignored.

  <time> —

- -collapsedelta | -collapsetime | -nocollapse

  (optional) Controls preservation of events in the resulting WLF file. The data preserved depends on how events were recorded in the input WLF file. Specifying a finer granularity of preservation than the input WLF file will have no additional affect.

  -collapsedelta — (default) Preserves only the values at the end of a delta.

  -collapsetime — Preserves only the values at the end of a time step.

  -nocollapse — Preserves all events.

- -compress | -nocompress

  (optional) Controls compression of the output WLF file.

  -compress — Enables compression. (default)

-nocompress — Disables compression.

- -end <time>

  (optional) Specifies the simulation time at which filtering of <source_wlffile> is stopped and no further data is logged.

- -f

  (optional) Repeat status update every 10 seconds of real time unless an alternate time interval is specified with -i <intervalTime>.

- -f <object_list_file>

  (optional) Specifies an object_list_file created by wlfman items or by the user to include in <out_wlffile>.

  For user created object list files, the object list can include stream name, stream array name, or sub stream with a full path. All

- -i <intervalTime>

  (optional) Specifies the time delay before the next status update where the default is 10 seconds of real time if not specified.

  <intervalTime> — Any positive integer.

- -index | -noindex

  (optional) Controls indexing when writing the output WLF file. Indexing makes viewing wave data faster, however performance during optimization will be slower because indexing and optimization require significant memory and CPU resources. Disabling indexing makes viewing wave data slower unless the display is near the start of the WLF file. Disabling indexing also disables optimization of the WLF file but may provide a significant performance boost when archiving WLF files. Indexing and optimization information can be added back to the file using the wlfman optimize command.

  -index — Enables indexing. (default)

  -noindex — Disables indexing and optimization.

- -n

  (optional) Lists regions only (no signals).

- -nowarn <number>

  (optional) Selectively disables a category of warning messages.

  1 — Disables "Skipping unsupported object" warning message.

- -opt | -noopt

  (optional) Controls optimization of the output WLF file.

  -opt — Enables WLF file optimization. (default)

  -noopt — Disables WLF file optimization.

- -p <endTime>

  (optional) Specifies the simulation time at which wlfman will stop monitoring the WLF file.

  <endTime> — Any positive integer.

- -q

  (optional) Suppress normal status messages while monitoring.

- -r <object>

  (optional) Specifies an object (region) to recursively include in the output. If <object> is a signal, the output would be the same as using -s.

- -rank

  (optional) Sorts the wlfman profile report by percentage of the total file space used by each signal.

- -s <symbol>

  (optional) Specifies an object to include in the output. By default all objects are included.

- -t <resolution>

  (optional) Specifies the time resolution of the new WLF file. By default the resolution is the same as the source WLF file.

- -top <number>

  (optional) Filters the wlfman profile report so that only the top <number> signals in terms of file space percentage are displayed.

- -v

  (optional) Produces verbose output that lists the object type next to each object.

Examples

- Specifying the command:

  wlfman profile -rank top_vh.wlf

  Returns:

```
#Repeated ID #'s mean those signals share the same
#space in the wlf file.
#
# ID     Transitions   File %   Name
#-----   -----------   ------   ------------------------------------
    1          2192      33 %    /top_vh/pdata
    1                            /top_vh/processor/data
    1                            /top_vh/cache/pdata
    1                            /top_vh/cache/gen__0/s/data
    1                            /top_vh/cache/gen__1/s/data
    1                            /top_vh/cache/gen__2/s/data
    1                            /top_vh/cache/gen__3/s/data
    2          1224      18 %    /top_vh/ptrans
    3          1216      18 %    /top_vh/sdata
    3                            /top_vh/cache/sdata
    3                            /top_vh/memory/data
    4           675      10 %    /top_vh/strans
    5           423       6 %    /top_vh/cache/gen__3/s/data_out
    6           135       3 %    /top_vh/paddr.
.
.
.
```

- Specifying the command:

  wlfman profile -top 3 trans.wlf

  Returns:

```
#The following table lists the number of transitions and approximate
#wlf file space consumed (prior to compression) for each signal
#logged in the wlf file.
#Repeated ID #'s mean those signals share the same
#space in the wlf file.
#
# ID     Transitions   File %   Name
# -----   -----------   ------   -------------------------------
#     1          1001      11 %    /top/t3
#     2          1001      11 %    /top/t1
#     3             1       0 %    /top/s2
#The following table lists the number of transactions and
#approximate wlf file space consumed (prior to compression) for each
#stream or stream array logged in the wlf file.
#
# ID     Transactions   File %   Name
# -----   -----------   ------   -------------------------------
#     1     3000          61 %      /top/stream2
#     1     1000          17 %      /top/stream1
```

- Specifying the command:

  wlfman monitor -f -p 100000000 vsim.wlf

  Returns:

```
Setting end time to 100000000, measuring progress %
File    File     Percent
State   Time     Complete
open    7239185  7.2%
```

```
open    7691785  7.7%
open    8144385  8.1%
open    8596625  8.6%
```

Related Topics

- Recording Simulation Results With Datasets
- WLF File Parameter Overview

wlfrecover

This command attempts to "repair" WLF files that are incomplete due to a crash or if the file was copied prior to completion of the simulation. Use this command if you receive a “bad magic number” error message when opening a WLF file. You can run the command from the VSIM> or Questa SIM> prompt or from a shell.

Syntax

wlfrecover <filename> [-force] [-q]

Arguments

- <filename>

  (required) Specifies the WLF file to repair.

- -force

  (optional) Disregards file locking and attempts to repair the file.

- -q

  (optional) Hides all messages unless there is an error while repairing the file.

Related Topics

- Saving a Simulation to a WLF File

write cell_report

This command writes to the Transcript window or to a file, a list of Verilog modules which qualified for and received gate-level cell optimizations. Gate-level cell optimizations are applied at the module level, in addition to normal Verilog optimizations, to improve performance of gate-level simulations.

Syntax

write cell_report [-filter <number>] [-infile <filename>] [-nonopt] [[-outfile] <filename>

Arguments

- -filter <number>

  (optional) Excludes cells with instance counts fewer than <number>.

  <number> — Specified as an integer greater than 0.

- -infile <filename>

  (optional) Specifies a previously generated write report file to use as input. If not specified then the write report command will be run.

  <filename> — A saved write report.

- -nonopt

  (optional) Reports only non-optimized instances.

- [-outfile] <filename>

  (optional) Writes the report to a specified output file rather than the Transcript window.

  <filename> — Specifies the name of the file that will be saved.

Related Topics

- Reports for Gate-Level Optimizations

write format

This command records the names and display options of the HDL objects currently being displayed in the Analysis, List, Memory, Message Viewer, Test Browser, and Wave windows.

The file created is primarily a list of add list, add wave, and configure commands, though a few other commands are included (refer to "Output" below).

This file may be invoked with the do command to recreate the window format on a subsequent simulation run (refer to restart below.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

write format {<window_type>**}** [-window <window_name>] <filename>

write format restart [<option option1 …>] <filename>

Arguments

- <window_type>

  (required unless specifying restart) Specifies that the contents of the designated window are recorded in the file specified by <filename>.

  assertions — Records objects of the Assertions window.

  breakpoints — Records file line and signal breakpoints.

  coverdirective — Records objects of the Coverdirectives window.

  export_hier_config — Records hierarchical sort order for objects in the Verification Results Analysis window.

  list — Records objects of the List window.

  memory — Records objects of the Memory window.

  msgviewer — Records objects of the Message Viewer window.

  testbrowser — Records objects of the Verification Management Browser window.

  watch — Records objects of the Watch window.

  wave — Records objects of the Wave window.

  restart — Records objects of all windows and breakpoints in the *.do* file.

- restart

  (required) Creates a *.do* file that recreates all debug windows, all file/line breakpoints, and all signal breakpoints created using the when command. If the ShutdownFile *modelsim.ini* variable is set to this *.do* filename, it will call the write format restart command upon exit.

  When you load a format file, Questa SIM verifies the existence of the datasets required by that file. Questa SIM displays an error message if the requisite datasets do not all exist. To

force the execution of the format file even if all datasets are not present, use the -force switch with your do command. For example:

```
VSIM> do format.do -force
```

Note

Note that using the -force switch when datasets are not present will result in error messages for signals referencing the nonexistent datasets. Also, -force is recognized by the format file not the do command.

- <option option1 …>

  (optional) Excludes a specific type of information from write format restart *.do* file.

  -nobreak — Do not record breakpoints.

  -nolastnow — Do not report last now value.

  -nolist — Do not record the List window format.

  -nomemory — Do not record Memory window views.

  -nosource — Do not record source files.

  -novsim — Do not record the vsim command.

  -nowave — Do not record the Wave window format.

- -window <window_name>

  (optional) Specifies the List or Wave window for which you want contents recorded. Use when you have more than one instance of the List or Wave window.

- <filename>

  (required) Specifies the name of the output file where the data is to be written. You must specify the *.do* extension.

Examples

- Save the current data in the List window in a file named *alu_list.do*.

  ```
  write format list alu_list.do
  ```

- Save the current data in the Wave window in a file named *alu_wave.do*.

  ```
  write format wave alu_wave.do
  ```

- Below is an example of a saved Wave window format file.

```
onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate -format Logic /cntr_struct/ld
add wave -noupdate -format Logic /cntr_struct/rst
add wave -noupdate -format Logic /cntr_struct/clk
add wave -noupdate -format Literal /cntr_struct/d
add wave -noupdate -format Literal /cntr_struct/q
TreeUpdate [SetDefaultTree]
quietly WaveActivateNextPane
add wave -noupdate -format Logic /cntr_struct/p1
add wave -noupdate -format Logic /cntr_struct/p2
add wave -noupdate -format Logic /cntr_struct/p3
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {0 ns}
WaveRestoreZoom {0 ns} {1 us}
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -signalnamewidth 0
configure wave -justifyvalue left
```

In the example above, five signals are added with the *-noupdate* argument to the default window. The TreeUpdate command then refreshes all five waveforms. The second WaveActivateNextPane command creates a second pane which contains three signals.The WaveRestoreCursors command restores any cursors you set during the original simulation, and the WaveRestoreZoom command restores the Zoom range you set. These four commands are used only in saved Wave format files; therefore, they are not documented elsewhere.

Related Topics

- add list
- add wave
- configure

write list

This command records the contents of the most recently opened or specified List window in a list output file.

This file contains simulation data for all HDL objects displayed in the List window: VHDL signals and variables and Verilog nets and registers.

Arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

write list [-events] [-window <wname>] <filename>

Arguments

- -events

  (optional) Specifies to write print-on-change format where the default is tabular format.

- -window <wname>

  (optional) Specifies an instance of the List window that is not the default. The default List window is used if this argument is unspecified. Use the view command to change the default window.

  <wname> — The name of a List window that is not the default.

- <filename>

  (required) Specifies the name of the output file where the data is to be written.

Examples

- Save the current data in the default List window in a file named *alu.lst*.

```
write list alu.lst
```

- Save the current data in window ‘list1’ in a file named *group1.list*.

```
write list -win list1 group1.list
```

Related Topics

- write tssi

write preferences

This command saves the current GUI preference settings to a Tcl preference file. Settings saved include Wave, Objects, and Locals window column widths; Wave, Objects, and Locals window value justification; and Wave window signal name width.

Syntax

write preferences <preference file name>

Arguments

- <preference file name>

  (required) Specifies the name for the preference file. If the file is named *modelsim.tcl*, Questa SIM will read the file each time vsim is invoked. To use a preference file other than *modelsim.tcl* you must specify the alternative file name with the MODELSIM_TCL environment variable.

You can modify variables by editing the preference file with the Questa SIM notepad:

```
notepad <preference file name>
```

write report

This command prints a summary of the design being simulated including a list of all design units (VHDL configurations, entities, and packages, and Verilog modules) with the names of their source files. The summary includes a list of all source files used to compile the given design.

The Simulation Report contains the following information:

- Design Simulated — directory path of the design's top-level module
- Number of signals/nets in the design
- Number of processes in the design
- Simulator Parameters, including:
  - Current directory
  - Project file directory
  - Simulation time resolution
- List of design units used, including:
  - Module name
  - Architecture, if applicable
  - Library directory
  - Source file
  - Timescale
  - Occurrences

Some arguments to this command are order-dependent. Please read through the argument descriptions for more information.

Syntax

write report [-capacity [-l | -s] [-assertions | -classes | -cvg | -qdas | -solver]] | [-l | -s] | [-solver] | [-tcl] | [<filename>]

Arguments

- -capacity

  (optional) Reports data on memory usage of various types of SystemVerilog constructs in the design.

  Questa SIM collects memory usage data for assertions, classes, covergroups, dynamic objects, and the solver.

Each of these design object types has a switch that you can specify along with -capacity in order to display its memory data.

Must be specified first when specifying -assertions, -classes, -cvg, -qdas.

To display memory data for all object types, specify -capacity -l.

- -assertions

  (optional) Reports memory usage data for SystemVerilog assertions and cover directives. You must precede this argument with -capacity.

- -classes

  (optional) Reports memory usage data for the current number of objects allocated, the current memory allocated for class object, the peak memory allocated and peak time. You must precede this argument with -capacity.

- -cvg

  (optional) Reports memory usage data for the number of covergroups, cross, bins and memory allocated. You must precede this argument with -capacity.

- <filename>

  (optional) Specifies the name of the output file where the data is to be written. If <filename> is omitted, the report is written to the Transcript window.

- -l

  (optional) Generates more detailed information about the design, including a list of sparse memories or the memory capacity for all object types. You must precede this argument with -capacity when specifying a capacity report.

- -qdas

  (optional) Reports memory usage data for queues, dynamic arrays, and associative arrays. You must precede this argument with -capacity when specifying a capacity report.

- -s

  (optional) Generates a short list of design information. You must precede this argument with -capacity when specifying a capacity report.

- -solver

  (optional) When specified with capacity, -solver reports memory usage data for calls to randomize() and memory usage. You must precede this argument with -capacity.

  When specified alone, -solver returns information about the memory allocated by the constraint solver and returns statistics for every SystemVerilog randomize() call site by filename and line number.

- -tcl

  (optional) Generates a Tcl list of design unit information. This argument cannot be used with a filename.

Examples

- Save information about the current design in a file named *alu_rpt.txt*.

  write report alu_rpt.txt

- Display a short list of information regarding the memory capacity for covergroups in the design during the simulation so far.

  write report -capacity -s cvg

- Display information on all of the calls to randomize() made during simulation so far, along with the memory usage of those calls, number of calls, and callsite information.

  write report -capacity -solver

  returns:

```
CAPACITY REPORT Generated on Tue Jan 03 13:57:26 2012
#
# Total Memory Allocated:30.2M
# Reporting total capacity data for Solver
#      Calls    CurrMem    PeakMem    PeakTime@ns
# ---------- ---------- ---------- --------------
#        156       1.1M       1.3M            50
```

- Create a Simulation Report for the current simulation

  write report -l

  returns:

```
##
## SIMULATION REPORT          Generated on Mon Aug 10 12:56:15 2009
##
##
## Design simulated: <directory>\work.top(fast)
## Number of signals/nets in design: 89
## Number of processes in design: 74
##
## Simulator Parameters:
##
##     Current directory: <directory>\
##     Project file: <directory>\win32/../modelsim.ini
##     Simulation time resolution: 1ns
##
## List of Design units used:
##
##     Module: top
##     Architecture: fast
##     Library: <directory>\work
##     Source File: top.v
##     Timescale: 1ns / 1ns
##     Occurrences: 1
##
##     Module: proc
##     Architecture: fast
##     Library: <directory>\work
```

```
##     Source File: proc.v
##     Timescale: 1ns / 1ns
##     Occurrences: 1

...
```

write timing

This command displays path delays and timing check limits, unadjusted for delay net delays, for the specified instance.

When the write timing command reports interconnect delays on a Verilog module instance you will see either MIPDs (Module Input Port Delays) or MITDs (Module Transport Port Delays) reported. If you specify either the +multisource_int_delays or the +transport_int_delays switch with the vsim command, INTERCONNECT delays will be reported as MITDs. Otherwise they will be reported as MIPDs. An MIPD report may look like the following:

```
# /top/u1: [mymod:src/5/test.v(18)]
#   MIPD(s):
#     Port clk_in: (6, 6, 6)
```

An MITD report may look like the following:

```
# /top/u1: [mymod:src/5/test.v(18)]
#   MITDs to port clk_in:
#     From port /top/p/y = (6)
```

Syntax

write timing [-recursive] [<filename>] [<instance_name1>…<instance_nameN>] [-simvalues]

Arguments

- -file <filename>

  (optional) Specifies the name of the output file where the data is to be written. If the -file argument is omitted, timing information is written to the Transcript window.

  <filename> — Any valid filename. May include special characters and numbers.

- <instance_name1>…<instance_nameN>

  (required) The name(s) of the instance(s) for which timing information will be written. If <instance_name> is omitted, the command returns nothing.

- -recursive

  (optional) Generates timing information for the specified instance and all instances underneath it in the design hierarchy.

- -simvalues

  (optional) Displays optimization-adjusted values for delay net delays.

Examples

- Write timing about */top/u1* and all instances underneath it in the hierarchy to the file *timing.txt*.

  write timing -r -f timing.txt /top/u1

- Write timing information about the designated instances to the Transcript window.

  write timing /top/u1 /top/u2 /top/u3 /top/u8

write transcript

This command writes the contents of the Transcript window to the specified file. The resulting file can then be modified to replay the transcribed commands as a DO file (macro).

The command cannot be used in batch mode. In batch mode use the standard Transcript file or redirect stdout.

Syntax

write transcript [<filename>]

Arguments

- <filename>

  (optional) Specifies the name of the output file where the data is to be written. If the <filename> is omitted, the transcript is written to a file named *transcript*.

Related Topics

- do
- Saving a Transcript File as a DO file

write tssi

This command records the contents of the default or specified List window in a "TSSI format" file.

The file contains simulation data for all HDL objects displayed in the List window that can be converted to TSSI format (VHDL signals and Verilog nets). A signal definition file is also generated.

The List window needs to be using symbolic radix in order for write tssi to produce useful output.

Syntax

write tssi [-window <wname>] <filename>

Arguments

- -window <wname>

  (optional) Specifies an instance of the List window that is not the default. Otherwise, the default List window is used. Use the view command to change the default window.

- <filename>

  (required) Specifies the name of the output file where the data is to be written.

Description

If the <filename> has a file extension (e.g., *listfile.lst*), then the definition file is given the same file name with the extension *.def* (e.g., *listfile.def*). The values in the listfile are produced in the same order that they appear in the List window. The directionality is determined from the port type if the object is a port, otherwise it is assumed to be bidirectional (mode INOUT).

Objects that can be converted to SEF are VHDL enumerations with 255 or fewer elements and Verilog nets. The enumeration values U, X, 0, 1, Z, W, L, H and - (the enumeration values defined in the IEEE Standard 1164 std_ulogic enumeration) are converted to SEF values according to the table below. Other values are converted to a question mark (?) and cause an error message. Though the write tssi command was developed for use with std_ulogic, any signal which uses only the values defined for std_ulogic (including the VHDL standard type bit) will be converted.

std_ulogic State Characters	**SEF State Characters**		
	Input	**Output**	**Bidirectional**
U	N	X	?
X	N	X	?
0	D	L	0
1	U	H	1
Z	Z	T	F

std_ulogic State Characters	SEF State Characters		
	Input	**Output**	**Bidirectional**
W	N	X	?
L	D	L	0
H	U	H	1
-	N	X	?

Bidirectional logic values are not converted because only the resolved value is available. The TSSI TDS ASCII In Converter and ASCII Out Converter can be used to resolve the directionality of the signal and to determine the proper forcing or expected value on the port. Lowercase values x, z, w, l, and h are converted to the same values as the corresponding capitalized values. Any other values will cause an error message to be generated the first time an invalid value is detected on a signal, and the value will be converted to a question mark (?).

Note

The TDS ASCII In Converter and ASCII Out Converter are part of the TDS software. Questa SIM outputs a vector file, and TSSI tools determine whether the bidirectional signals are driving or not.

Related Topics

- tssi2mti

write wave

This command records the contents of the currently active or specified Wave window in PostScript format.

The output file can then be printed on a PostScript printer.

Syntax

write wave **<filename>** [-end <time>] [-landscape] [-height <real_num>] [-margin <real_num>] [-perpage <time>] [-portrait] [-start <time>] [-width <real_num>] [-window <wname>]

Arguments

- <filename>

  (required) Specifies the name of the PostScript *(.ps)* output file.

- -end <time>

  (optional) The simulation time at which the record will end.

  <time> — Specified as a positive integer or decimal number where the units are the current simulation time resolution.

- -height <real_num>

  (optional) Specifies the paper height in inches.

  <real_num> — Specified as a positive integer or decimal number where the default is 11.0.

- -landscape

  (optional) Use landscape (horizontal) orientation. (default)

- -margin <real_num>

  (optional) Specifies the margin in inches.

  <real_num> — Specified as a positive integer or decimal number where the default is 0.5.

- -perpage <time>

  (optional) Specifies the time width per page of output.

  <time> — Specified as a positive integer or decimal number where the units are the current simulation time resolution.

- -portrait

  (optional) Use portrait (vertical) orientation where the default is landscape (horizontal).

- -start <time>

  (optional) Specifies the start time to be written.

<time> — Specified as a positive integer or decimal number where the units are the current simulation time resolution.

- -width <real_num>

  (optional) Specifies the paper width in inches.

  <real_num> — Specified as a positive integer or decimal number where the default is 8.5.

- -window <wname>

  (optional) Specifies an instance of the Wave window that is not the default. Otherwise, the default Wave window is used. Use the view command to change the default window.

Examples

- Save the current data in the Wave window in a file named *alu.ps*.

  write wave alu.ps

- Save the current data in window ‘wave2’ in a file named *group2.ps*.

  write wave -win wave2 group2.ps

- Write two separate pages to *top.ps*. The first page contains data from 600ns to 700ns, and the second page contains data from 701ns to 800ns.

  write wave -start 600ns -end 800ns -perpage 100ns top.ps

  To make the job of creating a PostScript waveform output file easier, use the **File > Print Postscript** menu selection in the Wave window.

xml2ucdb

The xml2ucdb is a utility used to convert an XML testplan file to a *.ucdb* file. The configuration settings for this utility are read automatically from the *xml2ucdb.ini* file, located in *<install_dir>/vm_src/* directory.

The settings specified by this command override any settings in the *xml2ucdb.ini* file (see "xml2ucdb.ini Configuration File").

For information about the XML language, see the "XML 1.0 Specification" available on the web.

The xml2ucdb command generates a return value of 0 (no errors or warnings), 1 (*.ucdb* was generated with errors and/or warnings), or 2 (*.ucdb* file not generated) upon command completion. When used within scripts, this returned value can be used to determine what actions should be taken next, depending on the success of the command.

Syntax

xml2ucdb [<options>] **<XML_filename>** {**<ucdb_filename>** | **-ucdbfilename <file>**}

Where <options> are:

[-help]
[-autonumber | -noautonumber] [-autoweight]
[-cfgfilename <path/ini_file_name>] [-createcovitem]
[-datafields <str>] [-datalabels <str>] [-datatags <tag>] [-debug] [-descriptiontag <tag>]
[-dofilename <file>] [-excelsheet <sheet_name>]
[-excludesections {<sec>[.<sec>][-<sec>[.<sec>]}[,<...>]] [-excludetags <tags>]
[-format {<format>}] [-formatlist]
[-G<varname>=<value>] [-goaltag <tags>] [-inherit] [-linktag <tag>]
[-linkattr <tag>] [-modelsimini <path/modelsim.ini>] [-root <str>] [-sectionprefix <str>]
[-sectiontags <tag>] [-searchpath <path/to/XML_input>] [-startsection <num>]
[-startstoring <num>] [-starttags <tags>] [-stylesheet <file>] [-stoptags <tags>]
[-tagprefix <str>] [-tagseparators <str>] [-title <str>] [-titletag <tag>] [-typeattr <name>]
[-varfile <path/varfile_name>] [-verbose] [-version] [-viewtags] [-viewall] [-weighttag <tags>]

Arguments

- -autonumber | -noautonumber

  (optional) Enables (-autonumber) or disables (-noautonumber) the automatic generation of testitem numbers from section tags. By default, the autonumbering is disabled. Use -autonumber to enable it. If you override the default off setting by enabling autonumbering in a custom configuration (*xml2ucdb.ini*) file, use -noautonumber to subsequently turn if off.

- -autoweight

  (optional) Enables the autoweighting method of coverage calculation for testplan scopes and coverage items that are linked to a testplan. By default, autoweighting is disabled. For more

detailed information on the autoweight and default (non-autoweighted) methods of coverage calculation, see Coverage Calculation in Testplans.

- -createcovitem

  (optional) Creates a single bin for each unlinked item (a link in the testplan which is not found in the design) when either the Unimplemented column is not present in the testplan, or the corresponding link doesn't have any entry in the Unimplemented column. If an integer or "No" is specified in the Unimplemented column, that specification is honored. For more information, see "Counting the Coverage Contribution from Unimplemented Links".

- -cfgfilename <path/ini_file_name>

  (optional) Specify an alternative XML2UCDB configuration file to use (i.e.: an INI file). If the file does not exist, the XML2UCDB utility searches: (a) the current $cwd, and (b) $MODELTECH/vm_src. The <path/ini_file_name> argument is a complete filename with an optional path, such as:

  /dir2/xml2ucdb.ini

- -datafields <str>

  (optional, may be overridden by -format) Specifies data fields, in the exact order that the columns appear in the testplan UCDB being imported. The -datafields argument uses a set of pre-defined keywords, each of which are treated by QuestaSim as a kind of "reserved" word. These case sensitive, order dependent, reserved words should be identical to those listed in the table of default expected columns/fields within section"The Fields in the Verification Plan". You can use these words in the configuration file, separated by the character specified with -tagseparator, in order to specify your own testplan's columns/fields. This switch can also be used to add additional, user-defined data fields to the testplan. Blank fields in XML source can be specified with the null character ("-").

  For more information on the datafields included the default *xml2ucdb.ini* file, located in <install_dir>/vrm_src/, refer to "Parameter for Mapping by Column Sequence".

- -datalabels <str>

  (optional) Specifies labels for the data fields.

- -datatags <tag>

  (optional) Specifies XML tag for item data fields.

- -debug

  (optional) Prints out internal debug information.

- -descriptiontag <tag>

  (optional) Specifies XML tag or tag list for description fields.

- -dofilename <file>

  (optional) Creates a .do file that contains coverage CLI commands used to link testplan with coverage items. This is also known as the testplan mapping file. The testplan mapping file

contains the coverage tag commands necessary to tag and link the coverage objects to the testplan items. The default: no mapping file.

- -excludesections {<sec>[.<sec>][-<sec>[.<sec>]}[,<...>]

  (optional) Specifies section(s) in the source plan to exclude from the generated testplan UCDB. The format of the sections supports either single or range values, or a combination of both, separated by commas ‘,’. For example, the command:

  xml2ucdb <other_args> -excludesections 1.2,1.4-1.7,2.2-4

  excludes sections 1.2, 1.4, 1.5, 1.6, 1.7, 2.2, 2.3, and 2.4. Excludes all children of excluded sections as well.

- -excludetags <tags>

  (optional) Specifies XML tag or tag list for tags to exclude from the processing.

- -excelsheet <sheet_name>

  (optional) Imports data from one specific sheet in an Excel spreadsheet. <sheet_name> is the exact string as it appears in the tab (“Sheet1”, “Sheet2”, etc.) at the bottom of an Excel spreadsheet.

- -format {<format>}

  (optional) Specifies the format to be used for the command, from a list of formats (pre-defined XML semantic definitions) listed in the *xml2ucdb.ini* file, located in the *<install_dir>/vm_src* directory.

  Essentially, this command is a shortcut for specifying the necessary parameters for successful testplan extraction. Within the *xml2ucdb.ini* file for each of the accepted formats (Excel, Word, etc.), there is a set of extraction parameters. The -format argument acts as a shortcut, telling xml2ucdb to use that specific pre-defined set of extraction parameters. Using -format, you do not need to define all the parameters specific to a particular format within the xml2ucdb command itself (-datafields, -datalabels, -datatags, etc.). For more details, see the section “Verification Plan Contents”.

  You can retrieve a list of valid <format> options to specify by running:

  xml2ucdb -formatlist

- -formatlist

  (optional) Returns the currently defined formats (pre-defined XML semantic definitions in Tcl) listed in the *xml2ucdb.ini* file located in the *<install_dir>/vm_src* directory.

  By default, the formats are: Excel, Word, DocBook, Frame, GamePlan, and MVCExcel (an MVC (Questa Verification IP) related format).

- -G<varname>=<value>

  (optional) Assigns <value> to variable <varname>. Allows re-use of instance based plans. These values override any set in the file named with -varfile <file>. If no value is specified for a defined variable, an error message results and the UCDB fails to generate. No whitespace should exist between -G and <varname>, and if found, an error will result. Any

spaces found in <varname> are removed; however spaces in the <value> are valid and taken into account in the replaced value. For example, the spaces in the entry `-Gvar1= val 1` is valid: the variable *var1* is replaced with the value " *val 1"*. For more information, see "Parameterizing Testplans".

- -goaltag <tags>

  (optional) Specifies the XML tag or tag list for a goal field.

- -help

  (optional) Prints Help Message.

- -inherit

  (optional) Used with a nested testplan. Causes storing state (resulting from start/stop tags or startstoring tag) to be inherited by the nested testplan.

- -linkattr <tag>

  (optional) Specifies XML tag attribute for cover items.

- -linktag <tag>

  (optional) Specifies XML tag or tag list for cover items.

- -modelsimini <path/modelsim.ini>

  (optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified in the MODELSIM environment variable. Specifies either an absolute or relative path to the initialization file, including the file itself. On Windows systems, the path separator should be a forward slash (/).

- -root <str>

  (optional) Specifies the root name to be prepended to each section number (in non-autonumbered testplans, i.e. spreadsheets) of the testplan. Allows you to specify a root testplan, in which to "nest" another testplan, thereby creating a hierarchical testplan. If no -root specification is used, the current section number is assumed to be the root for the nested testplan.

- -searchpath <path/to/XML_input>

  (optional) Specifies the path where XML import file is to be found. If this switch is not specified for a hierarchical (nested) testplan, any existing setting for parent testplan(s) is inherited. If none in ancestor testplans, the setting in the *xml2ucdb.ini* file is used.

- -sectiontags <tag>

  (optional) Specifies XML tag or tag list for a test item section number (such as "tag1:tag2:tag3").

- -sectionprefix <str>

  (optional) Specifies the prefix for section numbering (such as "tag1prefix:tag2prefix:tag3prefix").

- -startsection <num>

  (optional) Sets starting item number: defaults 0.

- -startstoring <num>

  (optional) Specifies the Storing to begin at an item number: defaults 0.

- -starttags <tags>

  (optional) Specifies XML tag or tag list for a tag to start the processing.

- -stoptags <tags>

  (optional) Specifies XML tag or tag list for a tag to stop the processing.

- -stylesheet <file>

  (optional) Specifies the name of XSL pre-processing stylesheet to be used.

- -tagprefix <str>

  (optional) Specifies a string to be prefixed to UCDB tag names. For a top-level testplan, if the tagprefix is not set, the value specified with -title is used. If any whitespace is contained in the title, it is replaced with underscore characters for use as the tagprefix. For nested testplans, there is no default tagprefix: if you do not specify a tagprefix, none is used for the nested testplan.

- -tagseparators <str>

  (optional) Specifies a list of characters used as tag separators for tag arguments accepting multiple tags. Applies only to the taglist parameters (-starttags, -stoptags, excludetags, etc.) specified on the command line.

- -title <str>

  (optional) Specifies a string to be used as the title for the testplan. For a top-level testplan, if the title is not set, the basename of the input XML file is used. For nested testplans, there is no default title: if you do not specify a title, none is used for the nested testplan.

- -titletag <tag>

  (optional) Specifies XML tag or tag list for a test item name, or start tag for Data fields.

- -typeattr <name>

  (optional) Specifies an attribute containing the "type" of each coverage item (coverpoint, covergroup, etc.). Types are case-insensitive.

- <ucdb_filename>

  (required unless -ucdbfilename is specified) Specifies the name for the *.ucdb* output file.

- -ucdbfilename <file>

  (required unless <ucdb_filename> is specified) Specifies the name for the *.ucdb* output file.

- -verbose

  (optional) Prints the testplan hierarchy and design mapping. Also displays values being used for parameters, if any.

- -varfile <path/varfile_name>

  (optional) Name of a file containing the variables for substitution within plan. Include a path if the varfile is not contained in the directory in which the xml2ucdb command is issued. The <varfile_name> contains a list of <variable>=<value> entries, one per line. If spaces are found in a <variable>, they are ignored, whereas, spaces in <value> specifications are honored. The specifications made by this argument can be overridden with a -G<varname>=<value>. For more information, see "Parameterizing Testplans".

- -version

  (optional) Prints the version number of the utility.

- -viewall

  (optional) Prints both tags and text contents of XML File.

- -viewtags

  (optional) Prints tag contents of XML File.

- -weighttag <tags>

  (optional) Specifies the XML tag or tag list for a weight field.

- <XML_filename>

  (optional) Specifies the input verification plan XML file to be converted. The path can be a full or relative path to the file location. On Windows systems the path separator should be a slash (/), rather than a backslash (\), for example:

  ```
  C:/design/vplan/verification.xml
  ```

Examples

- Convert a user-customized XML file, specifying the required columns in the testplan, skipping a field (-), and adding a OWNER and RESPONSIBLE field:

  xml2ucdb -datafields "Section,Title,Description,OWNER, -,RESPONSIBLE,Type,Weight,Goal"

- Convert an Excel formatted XML file called *input.xml* to a UCDB format file, *output.ucdb*:

  xml2ucdb -format Excel input.xml output.ucdb

- Convert only a specified sheet, *Sheet1*, of an Excel formatted XML file called *input.xml* to a UCDB format file, *output.ucdb*:

  xml2ucdb -format Excel -excelsheet Sheet1 input.xml output.ucdb

Related Topics

- Importing an XML Verification Plan
- XML Terms / Concept Review for Plan Import
- xml2ucdb.ini Configuration File
- Parameterizing Testplans
- coverage goal
- coverage tag
- coverage unlinked
- coverage weight
- vcover attribute
- vcover merge
- vcover testnames

xprop assertlimit

This command is used with simulations run with vopt -xprop. Sometimes, the same x-propagated assertion can be triggered at every interval, unnecessarily slowing down the simulation and making it noisy. You can use the xprop assertlimit command to set the fail count limit <n> for these X-propogated assertions.

Syntax

xprop assertlimit <n>

Arguments

- <n>

  Specifies an integer value as the number of times an xprop assertion is triggered. Default is 5. To set the assertions to unlimited, set <n> as -1.

  This setting overrides any value set using the XpropAssertionLimit *modelsim.ini* variable.

Related Topics

- vopt -xprop argument
- X Propagation in Simulation
- XpropAssertionLimit *modelsim.ini* variable

xprop disable

The xprop disable command can be used on a design previously optimized using vopt -xprop. This command disables the xprop effect, which is a more pessimistic, gate-level simulation (GLS) mode of propagating Xs through a design. When disabled, the simulation is run with the default, more optimistic, RTL X-propagation.

Syntax

xprop disable

Arguments

None

Related Topics

- vopt -xprop argument
- X Propagation in Simulation

xprop enable

Use the xprop enable command to re-enable xprop functionality that has been previously disabled with the xprop disable command. You must have a design loaded from vopt that used the -xprop argument.

Syntax

xprop enable

Arguments

None

Related Topics

- vopt -xprop argument
- X Propagation in Simulation

Index

— Symbols —

’hasX, hasX, 38
+typdelays, 839, 902
$finish behavior, customizing, 954

— Numerics —

2001, keywords, disabling, 840

— A —

abort command, 68
absolute time, using @, 29
add atv command, 69
add button command, 70
add dataflow command, 73
add list command, 75
add log command, 402
add memory command, 80
add message command, 82
add schematic command, 84
add testbrowser command, 86
add watch command, 87
add wave command, 89
add_cmdhelp command, 97
add_menu command, 99
add_menucb command, 101
add_menuitem simulator command, 103
add_separator command, 105
add_submenu command, 106
addTime command, 565
alias command, 108
analog
    signal formatting, 91
annotating interconnect delays,
        v2k_int_delays, 961
archive load command, 109
archive write command, 110
argument, 826
arrays
    indexes, 17
    slices, 17
arrays, VHDL, searching for, 34
assertion action command, 111
assertion active command, 114
assertion count command, 116
assertion enable command, 118
assertion fail command, 121
assertion pass command, 123
assertion profile command, 125
assertion report command, 127
assertions
    count, 116
    enable, 118
    enabling, 121, 123
    failure behavior, 121
    pass behavior, 123
    profiling, 125
    reporting, 127
    testing for with onbreak command, 431
    thread viewing, 69, 129
assume directives
    -assume argument, 910
    -noassume argument, 925
attributes, of signals, using in expressions, 38
atv log command, 129
automatic saving of coverage, 906

— B —

batch_mode command, 131
batch-mode simulations
    halting, 1001
bd (breakpoint delete) command, 132
binary radix, mapping to std_logic values, 45
blackbox
bookmark add wave command, 134
bookmark delete wave command, 136
bookmark goto wave command, 137
bookmark list wave command, 138
bp (breakpoint) command, 139
break
    on signal value, 995

breakpoints
conditional, 995
continuing simulation after, 510
deleting, 132
listing, 139
setting, 139
signal breakpoints (when statements), 995
time-based
in when statements, 1001
bus contention checking, 158
configuring, 160
disabling, 161
bus float checking
configuring, 164
disabling, 165
enabling, 162
busses
user-defined, 95
buttons, adding to the Main window toolbar, 70

— C —

C callstack
moving down, 481
moving up, 442
C debugging, 151
call command, 146
case choice, must be locally static, 678
case sensitivity
VHDL vs. Verilog, 23
cd (change directory) command, 150
cdbg command, 151
change command, 154
change_menu_cmd command, 157
check contention add command, 158
check contention config command, 160
check contention off command, 161
check float add command, 162
check float config command, 164
check float off command, 165
check stable off command, 166
check stable on command, 167
-check_synthesis argument, 669
checkpoint command, 169
class instance garbage collector, 387, 389
class member selection, syntax, 17
class objects, viewing, 174, 175, 177, 179, 182, 184, 185, 187
classinfo command, 174, 175, 177, 179, 182, 184, 185, 187
Code Coverage
coverage analyze command, 233
coverage clear command, 247
coverage exclude command, 257
coverage goal command, 267
coverage report command, 281
coverage save command, 295
coverage tag command, 299
coverage testnames command, 306
coverage unlinked command, 307
coverage weight command, 311
merging reports, 707
toggle coverage
excluding signals, 571
vcover report command, 730
Color
radix, 493
example, 494
combining signals, busses, 95
command line args, accessing
vopt sc_arg command, 895
vsim sc_arg command, 962
commands
.main clear, 67
abort, 68
add atv, 69
add button, 70
add dataflow, 73
add list, 75
add memory, 80
add message, 82
add schematic, 84
add testbrowser, 86
add watch, 87
add wave, 89
add_cmdhelp, 97
add_menu, 99
add_menucb, 101
add_menuitem, 103
add_separator, 105
add_submenu, 106

alias, 108
archive load, 109
archive write, 110
assertion action, 111
assertion active, 114
assertion count, 116
assertion enable, 118
assertion fail command, 121
assertion pass, 123
assertion profile, 125
assertion report, 127
atv log, 129
batch_mode, 131
bd (breakpoint delete), 132
bookmark add wave, 134
bookmark delete wave, 136
bookmark goto wave, 137
bookmark list wave, 138
bp (breakpoint), 139
call, 146
cd (change directory), 150
cdbg, 151
change, 154
change_menu_cmd, 157
check contention add, 158
check contention config, 160
check contention off, 161
check float add, 162
check float config, 164
check float off, 165
check stable off, 166
check stable on, 167
checkpoint, 169
classinfo, 174, 175, 177, 179, 182, 184, 185, 187
compare add, 189
compare annotate, 195
compare clock, 197
compare configure, 199
compare continue, 201
compare delete, 202
compare end, 203
compare info, 204
compare list, 206
compare options, 207
compare reload, 212
compare reset, 213
compare run, 214
compare savediffs, 215
compare saverules, 216
compare see, 217
compare start, 214, 219
compare stop, 221
compare update, 222
configure, 223
context, 229
coverage analyze, 233
coverage attribute, 242
coverage clear, 247
coverage create, 249
coverage edit, 252
coverage exclude, 257
coverage goal, 267
coverage loadtestassoc, 271
coverage open, 272
coverage ranktest, 273
coverage report, 281
coverage save, 295
coverage tag, 299
coverage testnames, 306
coverage unlinked, 307
coverage weight, 311
dataset alias, 314
dataset clear, 315
dataset close, 316
dataset config, 317
dataset current, 319
dataset info, 320
dataset list, 321
dataset open, 322
dataset rename, 323
dataset restart, 324
dataset save, 325
dataset snapshot, 326
delete, 329
describe, 330
disable_menu, 333
disable_menuitem, 334
disablebp, 332
do, 335

down, 337
drivers, 339
dumplog64, 341
echo, 342
edit, 343
enable_menu, 345
enable_menuitem, 346
enablebp, 344
encoding, 347
environment, 348
examine, 350
exit, 358
fcover configue, 359
find, 362
find connections, 368
find drivers, 369
find infiles, 374
find insource, 375
force, 377
fsm list, 384
fsm properties, 385
fsm view, 386
gc configure, 387
gc run, 389
gdb dir, 390
getactivecursortime, 391
getactivemarkertime, 392
help, 393
history, 394
jobspy, 395
layout, 396
lecho, 398
left, 399
log, 402
lshift, 406
lsublist, 407
mem compare, 408
mem display, 409
mem list, 412
mem load, 413
mem save, 417
mem search, 420
next, 423
noforce, 424
nolog, 425
notation conventions, 15
notepad, 427
noview, 428
nowhen, 429
onbreak, 430
onElabError, 432
onerror, 433
onfinish, 435
pa autotestplan, 436
pa msg, 437
pa report, 440
pause, 441
pop, 442
power add, 443
power report, 449
power reset, 452
printenv, 454, 455
process report, 456
profile clear, 457
profile interval, 458
profile off, 459
profile on, 460
profile open, 462
profile option, 463
profile reload, 465
profile report, 466
profile save, 470
profile summary, 472
property list, 477
property wave, 479
push, 481
pwd, 482
questasim, 483
quietly, 484
quit, 485
qverilog, 486
radix, 490
radix define, 492
radix list, 497
radix name, 498
readers, 500
report, 501
restart, 503
restore, 505
resume, 506

right, 507
run, 510
runStatus, 513
sccom, 515
scgenmod, 525
sdfcom, 528
search, 530
searchlog, 533
see, 536
seetime, 537
setenv, 539
shift, 540
show, 541
simstats, 542
simstatslist, 544
stack down, 546
stack frame, 547
stack level, 548
stack tb, 549
stack up, 550
status, 551
stop, 554
suppress, 555
tb (traceback), 549, 557
tcheck_set, 558
tcheck_status, 562
Time, 565
toggle add, 568
toggle disable, 571
toggle enable, 573
toggle report, 574
toggle reset, 576
tr color, 577
tr order, 580
tr uid, 582
transcribe, 584
transcript, 585
transcript file, 586
transcript path, 588
transcript sizelimit, 589
TreeUpdate, 1018
tssi2mti, 606
typespec, 607
unsetenv, 610
up, 611
uvm call, 613
uvm configtracing, 616
uvm displayobjections, 617
uvm findregisters, 620
uvm findsequences, 622
uvm handle, 624, 626
uvm printconfig, 627
uvm printfactory, 629
uvm printstreams, 631
uvm printtopology, 633
uvm setverbosity, 635
uvm simpath, 637
uvm traceobjections, 639
uvm uvmpath, 641
variables referenced in, 28
vcd add, 643
vcd checkpoint, 645
vcd comment, 646
vcd dumpports, 647
vcd dumpportsall, 650
vcd dumpportsflush, 651
vcd dumpportslimit, 652
vcd dumpportsoff, 653
vcd dumpportson, 654
vcd file, 655
vcd files, 657
vcd flush, 659
vcd limit, 660
vcd off, 661
vcd on, 662
vcom, 665
vcover attribute, 691
vcover dump, 701
vcover history, 704
vcover merge, 707
vcover parallelmerge, 716
vcover ranktest, 719
vcover remove, 728
vcover report, 730
vcover testnames, 757
vdel, 761
vdir, 763
vencrypt, 766
verror, 770
vgencomp, 772

vhencrypt, 774
view, 777
virtual count, 781
virtual define, 782
virtual delete, 783
virtual describe, 784
virtual expand, 785
virtual function, 786
virtual hide, 790
virtual log, 791
virtual nohide, 793
virtual nolog, 794
virtual region, 796
virtual save, 797
virtual show, 798
virtual signal, 799
vlib, 805
vlog, 808
vmake, 844
vmap, 846
vopt, 848
vsim, 906
vsimDate, 969
vsimId, 969
vsimVersion, 969
vsource, 971
wave, 972
wave create, 976
wave edit, 982
wave export, 986
wave import, 988
wave modify, 989
wave sort, 994
WaveActivateNextPane, 1018
WaveRestoreCursors, 1018
WaveRestoreZoom, 1018
when, 995
where, 1003
wlf2log, 1004
wlf2vcd, 1006
wlfman, 1007
wlfrecover, 1014
write cell_report, 1015
write format, 1016
write list, 1019
write preferences, 1020
write report, 1021
write timing, 1025
write transcript, 1027
write tssi, 1028
write wave, 1030
xml2ucdb, 1032
xprop assertlimit, 1039
xprop disable, 1040
xprop enable, 1041
commands formatTime, 383
comment characters in VSIM commands, 15
compare add command, 189
compare annotate command, 195
compare clock command, 197
compare configure command, 199
compare continue command, 201
compare delete command, 202
compare end command, 203
compare info command, 204
compare list command, 206
compare options command, 207
compare reload command, 212
compare reset command, 213
compare run command, 214
compare savediffs command, 215
compare saverules command, 216
compare see command, 217
compare start command, 214, 219
compare stop command, 221
compare update command, 222
compatibility, of vendor libraries, 763
compiling
range checking in VHDL, 685
SystemC, 515, 525
Verilog, 808
VHDL, 665
at a specified line number, 677
selected design units (-just eapbc), 676
standard package (-s), 685, 834
VHDL-2008
REAL_VECTOR, 676
compressing files
checkpoint files, 169
elaboration files, 913

VCD files, 647, 657
concatenation
directives, 43
of signals, 43
conditional breakpoints, 995
configurations, simulating, 906
configure command, 223
constants
in case statements, 678
values of, displaying, 330, 350
contention checking, 158
context command, 229
conversion
radix, 490
coverage
vcover testnames command, 757
coverage analylze command, 233
coverage attribute command, 242
coverage clear command, 247
coverage create command, 249
coverage edit command, 252
coverage exclude command, 257
coverage goal command, 267
coverage loadtestassoc command, 271
coverage open command, 272
coverage ranktest command, 273
coverage report command, 281
coverage save command, 295
coverage tag command, 299
coverage testnames command, 306
coverage unlinked command, 307
Coverage View mode
coverage loadtestassoc command, 271
coverage open command, 272
coverage weight command, 311
coverage, automatic save, 906
coverstore, auto-saved coverage, 906
customizing
adding buttons, 70

— D —

dataset alias command, 314
dataset clear command, 315
dataset close command, 316
dataset config command, 317
dataset current command, 319
dataset info command, 320
dataset list command, 321
dataset open command, 322
dataset rename command, 323
dataset restart command, 324
dataset save command, 325
dataset snapshot command, 326
datasets
environment command, specifying with, 348
declarations, hiding implicit with explicit, 689
default VOPT behavior, and .ini file, 848
+define+, 815
delay
interconnect, 924
+delay_mode_distributed, 816, 863
+delay_mode_path, 816, 863
+delay_mode_unit, 816, 863
+delay_mode_zero, 816, 863
’delayed, 38
delete command, 329
deltas
collapsing in WLF files, 941
dependencies, checking, 763
dependency errors, 673, 818
describe command, 330
design loading, interrupting, 906
design units
report of units simulated, 1021
Verilog
adding to a library, 808
directories
mapping libraries, 846
disable_menu command, 333
disable_menuitem command, 334
disablebp command, 332
dividers
adding from command line, 90
divTime ccommand, 565
do command, 335
DO file
executing, 335
DO files, 335
breakpoints, executing at, 141
forcing signals, nets, or registers, 377

parameters
passing, 335
relative directories, 335
shifting parameter values, 540
down command, 337
-dpiheader, vlog, 816, 948
-dpiheader, vopt, 864
drivers command, 339
dump files, viewing in the simulator, 663
dumplog64 command, 341

— E —

echo command, 342
edges, finding, 399, 507
edit command, 343
enable_menu command, 345
enable_menuitem command, 346
enablebp command, 344
encoding command, 347
encryption
+protect argument, 833
-nodebug argument (vcom), 680
-nodebug argument (vlog), 827
entities, specifying for simulation, 966
environment command, 348
environment variables
reading into Verilog code, 815
specifying UNIX editor, 343
state of, 455
using in pathnames, 22
environment, displaying or changing pathname, 348
eqTime command, 565
errors
getting details about messages, 770
onerror command, 433
SDF, disabling, 932
event order
changing in Verilog, 812, 815, 858
examine command, 350
exit command, 358
exiting the simulator, customizing behavior, 954
extended identifier, 37
extended identifiers, 23
extended toggle coverage, reporting, 284, 731, 734

— F —

fcover configue command, 359
file compression
checkpoint files, 169
elaboration files, 913
VCD files, 647, 657
find command, 362
find connections command, 368
find drivers command, 369
find infiles command, 374
find insource command, 375
fixed point radix, 492
floating point radix, 492
force
remove wire model, 927
force command, 377
foreign module declaration
Verilog example, 526
format file
List window, 1016
Wave window, 1016
formatTime command, 383, 566
fsm list command, 384
fsm properties command, 385
fsm view command, 386
Functional coverage
merging databases offline, 707
functional coverage
configuring directives, 359

— G —

gc configure command, 387
gc run command, 389
gdb dir command, 390
generics
assigning or overriding values with -g and -G, 917
examining generic values, 350
limitation on assigning composite types, 869, 918
getactivecursortime command, 391
getactivemarkertime command, 392
glitches

disabling generation
from command line, 944
global visibility
PLI/FLI shared objects, 919
shared objects, 869
gotolingk MGCGOTOVAR_user
DPI File Loading, 958
gteTime command, 565
gtTime command, 565
GUI_expression_format, 34
syntax, 36

— H —

'hasX, 38
hazards
-hazards argument to vlog, 820, 869
-hazards argument to vsim, 949
help command, 393
history
of commands
shortcuts for reuse, 32
history command, 394
HTML report
generating from .ucdb, 286, 737

— I —

implicit operator, hiding with vcom -explicit, 689
+incdir+, 820
interconnect delays, 924
annotating per Verilog 2001, 961
internal signals, adding to a VCD file, 643
interrupting design loading, 906
intToTime command, 565

— J —

jobspy command, 395

— K —

keywords
disabling 2001 keywords, 840
enabling SystemVerilog keywords, 836

— L —

layout command, 396
LD_LIBRARY_PATH, disabling default internal setting of, 925
lecho command, 398
left command, 399
+libcell, 824
libraries
dependencies, checking, 763
design libraries, creating, 805
listing contents, 763
refreshing library images, 685, 834
vendor supplied, compatibility of, 763
Verilog, 920
lint-style checks, 825
List window
adding items to, 75
loading designs, interrupting, 906
Local variables
print to Transcript, 122
log command, 402
log file
log command, 402
nolog command, 425
QuickSim II format, 1004
redirecting with -l, 920, 921
virtual log command, 791
virtual nolog command, 794
Log local variables, 122
lshift command, 406
lsublist command, 407
lteTime command, 565
ltTime command, 565

— M —

.main clear command, 67
master slave library (SystemC), including, 520
+maxdelays, 825, 876
mc_scan_plusargs, PLI routine, 955
mem compare command, 408
mem display command, 409
mem list command, 412
mem load command, 413
mem save command, 417
mem search command, 420
memory window
add memory command, 80
adding items to, 80
memory, comparing contents, 408
memory, displaying contents, 409

memory, listing, 412
memory, loading contents, 413
memory, saving contents, 417
memory, searching for patterns, 420
merge
    test-associated data, viewing, 233
merging coverage reports, 707
messages
    echoing, 342
    getting more information, 770
    loading, disabling with -quiet, 833, 895
    loading, disbling with -quiet, 684
-mfcu, 826
+mindelays, 826, 876
modelsim.ini
    default VOPT behavior, 848
mulTime command, 565
multi-source interconnect delays, 924

— N —

name case sensitivity, VHDL vs. Verilog, 23
ncsim, one step simulation, 486
negative pulses
    driving an error state, 956
neqTime command, 565
nets
    drivers of, displaying, 339
    readers of, displaying, 500
    stimulus, 377
    values of
        examining, 350
next command, 423
-no_risefall_delaynets, 952
-nodebug argument (vcom), 680
-nodebug argument (vlog), 827
noforce command, 424
+nolibcell, 824, 878
nolog command, 425
notepad command, 427
noview command, 428
+nowarn<CODE>, 830, 880
nowhen command, 429

— O —

object_list_file, WLF files, 1007
onbreak command, 430
onElabError command, 432
onerror command, 433
onfinish command, 435
optimizations
    disabling for Verilog designs, 831, 881
    disabling for VHDL designs, 683
    optimizing wlf files, 1008
    vopt command, 848
order of events
    changing in Verilog, 812, 815, 858

— P —

pa autotestplan command, 436
pa msg command, 437
pa report command, 440
parallel merge, vcover command, 716
parameters
    using with DO files, 335
pathnames
    in VSIM commands, 16
    spaces in, 16
pause command, 441
performance
    vopt command, 848
PLI
    loading shared objects with global symbol
        visibility, 919
pop command, 442
power add command, 443
Power Aware simulation, 928
power report command, 449
power reset command, 452
preference variables
    WildcardFilter, 24
Preoptimized Design Unit, 893
    and SDF file, 931
    maintain visibility into PDU, 893
    report Verilog modules, 858
printenv command, 454, 455
process report command, 456
profile clear command, 457
profile interval command, 458
profile off command, 459
profile on command, 460
profile open command, 462
profile option command, 463

profile reload command, 465
profile report command, 466
profile save command, 470
profile summary command, 472
projects
    override mapping for work directory with vcom, 521, 688
    override mapping for work directory with vlog, 488, 841
propagation, preventing X propagation, 926
property list command, 477
property wave command, 479
pulse error state, 956
push command, 481
pwd command, 482

— Q —

questasim command, 483
QuickSim II logfile format, 1004
quietly command, 484
quit command, 485
qverilog command, 486

— R —

Radix
    color, 493
        example, 494
radix
    display values in debug windows, 490
    of signals being examined, 77, 93, 353
    user defined, 492
radix command, 490
Radix define command
    setting radix color, 493, 494
radix define command, 492
    fixed point radix, 492
    floating point radix, 492
radix list command, 497
radix name command, 498
range checking
    disabling, 681
    enabling, 685
readers command, 500
RealToTime command, 566
record field selection, syntax, 17
refresh, dependency check errors, 673, 818
refreshing library images, 685, 834
report command, 501
reporting
    processes in the Process Window, 456
    variable settings, 28
resolution
    specifying with -t argument, 934
restart command, 503
restore command, 505
resume command, 506
right command, 507
run command, 510
runStatus command, 513

— S —

sc_stop()
    customizing simulator behavior, 954
scaleTime command, 566
sccom command, 515
-scdpidebug command, 962
scgenmod command, 525
-sclib command, 896, 962
scope resolution operator, 19
scope, setting region environment, 348
SCV library, including, 520
SDF
    annotation verbose mode, 933
    compiled SDF, 528
    controlling missing instance messages, 896, 932
    disabling individual checks, 558
    errors on loading, disabling, 932
    warning messages, disabling, 932
sdfcom command, 528
search command, 530
search libraries, 920
searching
    binary signal values in the GUI, 45
    List window
        signal values, transitions, and names, 34, 337, 611
    next and previous edge in Wave window, 399, 507
    VHDL arrays, 34
    Wave window

signal values, edges and names, 399, 507
searchlog command, 533
see command, 536
seetime command, 537
setenv command, 539
shared objects
loading with global symbol visibility, 869, 919
shift command, 540
shortcuts
command history, 32
command line caveat, 31
show command, 541
signals
alternative names in the Wave window (-label), 92
attributes of, using in expressions, 38
breakpoints, 995
combining into a user-defined bus, 95
drivers of, displaying, 339
environment of, displaying, 348
force time, specifying, 380
log file, creating, 402
pathnames in VSIM commands, 16
radix
specifying for examine, 77, 93, 353
readers of, displaying, 500
stimulus, 377
values of
examining, 350
simstats command, 542
simstatslist command, 544
simulating
delays, specifying time units for, 28
design unit, specifying, 906
ncsim style, 486
one step, 486
saving simulations, 402, 940
stopping simulation in batch mode, 1001
simulations
saving results, 325, 326
Simulator commands, 68
simulator resolution
vsim -t argument, 934
simulator version, 939, 969
simultaneous events in Verilog
changing order, 812, 815, 858
sml2ucdb command, 1032
source annotation, 519
spaces in pathnames, 16
sparse memories
listing with write report, 1022
specify path delays, 956
stability checking
disabling, 166
enabling, 167
stack down command, 546
stack frame command, 547
stack level command, 548
stack tb command, 549
stack up command, 550
startup
alternate to startup.do (vsim -do), 915
status command, 551
Std_logic
mapping to binary radix, 45
stop command, 554
subTime command, 565
suppress command, 555
synthesis
rule compliance checking, 669
SystemC
class and structure member naming syntax, 17
DPI, command for single-stepping across call boundaries, 962
master slave library, including, 520
specifying shared library path, command, 896, 962
verification library, including, 520
SystemVerilog
enabling with -sv argument, 836
multiple files in a compilation unit, 826
scope resolution, 19
SystemVerilog assertions
disabling, 926
SystemVerilog classes
call command, 146

— T —

tb command, 549, 557
tcheck_set command, 558
tcheck_status command, 562
Tcl
    history shortcuts, 32
    variable
        in when commands, 998
test management window
    adding UCDB files to, 86
test-associated data
    viewing, 233
testplan removal
    vcover remove command, 728
TFMPC
    disabling warning, 953
time
    absolute, using @, 29
    simulation time units, 28
time collapsing, 941
Time commands, 565
time resolution
    setting
        with vsim command, 934
time, time units, simulation time, 28
timescale directive warning
    disabling, 953
timing
    disabling checks, 829, 880
    disabling checks for entire design, 927
    disabling individual checks, 558
    status of individual checks, 562
title, Main window, changing, 936
toggle
    reporting extended, 284, 731, 734
toggle add command, 568
toggle coverage
    excluding signals, 571
    reporting, duplication of elements, 574
    reporting, ordering of nodes, 574
toggle disable command, 571
toggle enable command, 573
toggle report command, 574
toggle reset command, 576
toggle statistics
    enabling, 568
    reporting, 574
    resetting, 576
tr color command, 577
tr order command, 580
tr uid command, 582
transcribe command, 584
transcript
    clearing, 67
    redirecting with -l, 920, 921
transcript command, 585
transcript file command, 586
transcript path command, 588
transcript sizelimit command, 589
transitions, signal, finding, 399, 507
TreeUpdate command, 1018
TSCALE, disabling warning, 953
TSSI, 1028
tssi2mti command, 606
typespec command, 607

— U —

-u, 839
UCDB coverage
    coverage attribute command, 242
    vcover attribute command, 691
    vcover dump command, 701
    vcover history command, 704
undeclared nets, reporting an error, 825
Unified Power Format, 891
unsetenv command, 610
up command, 611
UPF (Unified Power Format), 891
user-defined bus, 95
User-defined radix, 492
uvm call command, 613
uvm configtracing command, 616
uvm displayobjections command, 617
uvm findregisters command, 620
uvm findsequencescommand, 622
uvm handle command, 624, 626
uvm printconfig command, 627
uvm printfactory command, 629
uvm printstreams command, 631
uvm printtopology command, 633
uvm setverbosity command, 635

uvm simpath command, 637
uvm traceobjections command, 639
uvm uvmpath command, 641

— V —

-v, 840
v2k_int_delays, 961
validTime command, 566
values
  describe HDL items, 330
  examine HDL item values, 350
variable settings report, 28
variables
  describing, 330
  referencing in commands, 28
  value of
    changing from command line, 154
    examining, 350
vcd add command, 643
vcd checkpoint command, 645
vcd comment command, 646
vcd dumpports command, 647
vcd dumpportsall command, 650
vcd dumpportsflush command, 651
vcd dumpportslimit command, 652
vcd dumpportsoff command, 653
vcd dumpportson command, 654
vcd file command, 655
VCD files
  adding items to the file, 643
  capturing port driver data, 647
  converting to WLF files, 663
  creating, 643
  dumping variable values, 645
  flushing the buffer contents, 659
  generating from WLF files, 1006
  inserting comments, 646
  internal signals, adding, 643
  specifying maximum file size, 660
  specifying name of, 657
  specifying the file name, 655
  state mapping, 655, 657
  turn off VCD dumping, 661
  turn on VCD dumping, 662
  viewing files from another tool, 663
vcd files command, 657
vcd flush command, 659
vcd limit command, 660
vcd off command, 661
vcd on command, 662
vcd2wlf command, 663
vcom command, 665
vcom Examples, 689
vcover attribute command, 691
vcover dump command, 701
vcover history command, 704
vcover merge command, 707
vcover parallelmerge command, 716
vcover ranktest command, 719
vcover remove command, 728
vcover report command, 730
vcover testnames command, 757
vdel command, 761
vdir command, 763
vector elements, initializing, 154
vencrypt command, 766
vendor libraries, compatibility of, 763
Verilog
  $finish behavior, customizing, 954
  capturing port driver data with -dumpports, 655
Verilog 2001
  disabling support, 840
verror command, 770
version
  obtaining with vsim command, 939
  obtaining with vsim<info> commands, 969
vgencomp command, 772
VHDL
  arrays
    searching for, 34
  binding, ignore default, 675
  compile
    1076-1987, 667
    1076-1993, 667
    1076-2002, 667
    1076-2008, 667
  conditions and expressions, automatic conversion of H and L., 679
  field naming syntax, 17
VHDL identifiers, perserving case, 526

VHDL-1987, enabling support for, 667
VHDL-1993, enabling support for, 667
VHDL-2002, enabling support for, 667
VHDL-2008
    package STANDARD
        REAL_VECTOR, 676
VHDL-2008, enabling support for, 667
vhencrypt command, 774
view command, 777
viewing
    waveforms, 940
virtual count commands, 781
virtual define command, 782
virtual delete command, 783
virtual describe command, 784
virtual expand commands, 785
virtual function command, 786
virtual hide command, 790
virtual log command, 791
virtual nohide command, 793
virtual nolog command, 794
virtual region command, 796
virtual save command, 797
virtual show command, 798
virtual signal command, 799
Visualizer Debug Environment
    vopt, 864
vlib command, 805
vlog
    multiple file compilation, 826
vlog command, 808
vmake command, 844
vmap command, 846
vopt
    path separator, 848
vopt command, 848
vsim
    disabling internal setting of
        LD_LIBRARY_PATH, 925
vsim build date and version, 969
vsim command, 906
vsim Examples, 967

— W —

warning message
    6846, 707
WARNING[8], -lint argument to vlog, 825
warnings
    SDF, disabling, 932
    suppressing VCOM warning messages,
        682, 830, 881
    suppressing VLOG warning messages,
        830, 880
    suppressing VSIM warning messages, 953
watch window
    add watch command, 87
    adding items to, 87
watching signal values, 87
wave commands, 972
wave create command, 976
wave cursor commands, 972
wave edit command, 982
wave export command, 986
wave import command, 988
wave log format (WLF) file, 940
    of binary signal values, 402
wave modify command, 989
wave sort command, 994
Wave window
    adding items to, 89
WaveActivateNextPane command, 1018
Waveform Comparison, 189
waveform editor
    creating waves, 976
    editing commands, 982
    importing vcd stimulus file, 988
    modifying existing waves, 989
    saving waves, 986
waveform logfile
    log command, 402
waveforms
    optimizing viewing of, 942
    saving and viewing, 402
WaveRestoreCursors command, 1018
WaveRestoreZoom command, 1018
when command, 995
when statement
    time-based breakpoints, 1001
where command, 1003
wildcard characters

for pattern matching in simulator commands, 23
WildcardFilter Preference Variable, 24
windows
List window
output file, 1019
saving the format of, 1016
Main window
adding user-defined buttons, 70
opening
from command line, 777
Wave window
path elements, changing, 227
WLF files
collapsing deltas, 941
collapsing time steps, 941
converting to VCD, 1006
creating from VCD, 663
indexing, 1008
limiting size, 942
log command, 402
merging, 1008
optimizing, 1008
optimizing waveform viewing, 942
repairing, 1014
saving, 325, 326
specifying name, 940
wlfman command, 1007
wlf2log command, 1004
wlf2vcd command, 1006
wlfman command, 1007
wlfrecover command, 1014
write cell_report command, 1015
write format command, 1016
write list command, 1019
write preferences command, 1020
write report command, 1021
write timing command, 1025
write transcript command, 1027
write tssi command, 1028
write wave command, 1030

— X —

X propagation
disabling for entire design, 926
disabling X generation on specific instances, 558
xprop assertlimit command, 1039
xprop disable command, 1040
xprop enable command, 1041

— Y —

-y, 841

— Z —

zoom
wave window
returning current range, 973

End-User License Agreement

The latest version of the End-User License Agreement is available on-line at:
www.mentor.com/eula

IMPORTANT INFORMATION

USE OF ALL SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF SOFTWARE INDICATES CUSTOMER'S COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.

END-USER LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Software (as defined in Section 2) and hardware (collectively "Products") between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If Customer does not agree to these terms and conditions, promptly return or, in the case of Software received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

1. **ORDERS, FEES AND PAYMENT.**

   1.1. To the extent Customer (or if agreed by Mentor Graphics, Customer's appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement (each an "Order"), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement, any applicable addenda and the applicable quotation, whether or not those documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order or presented in any electronic portal or automated order management system, whether or not required to be electronically accepted, will not be effective unless agreed in writing and physically signed by an authorized representative of Customer and Mentor Graphics.

   1.2. Amounts invoiced will be paid, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice. Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax, consumption tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer's sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer's behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.

   1.3. All Products are delivered FCA factory (Incoterms 2010), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics retains a security interest in all Products delivered under this Agreement, to secure payment of the purchase price of such Products, and Customer agrees to sign any documents that Mentor Graphics determines to be necessary or convenient for use in filing or perfecting such security interest. Mentor Graphics' delivery of Software by electronic means is subject to Customer's provision of both a primary and an alternate e-mail address.

2. **GRANT OF LICENSE.** The software installed, downloaded, or otherwise acquired by Customer under this Agreement, including any updates, modifications, revisions, copies, documentation and design data ("Software") are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors, who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form (except as provided in Subsection 5.2); (b) for Customer's internal business purposes; (c) for the term of the license; and (d) on the computer hardware and at the site authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Customer may have Software temporarily used by an employee for telecommuting purposes from locations other than a Customer office, such as the employee's residence, an airport or hotel, provided that such employee's primary place of employment is the site where the Software is authorized for use. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or services purchased, apply to the following: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be technically implemented through the use of authorization codes or similar devices); and (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions. For the avoidance of doubt, if Customer provides any feedback or requests any change or enhancement to Products, whether in the course of receiving support or consulting services, evaluating Products, performing beta testing or otherwise, any inventions, product improvements, modifications or developments made by Mentor Graphics (at Mentor Graphics' sole discretion) will be the exclusive property of Mentor Graphics.

3. **ESC SOFTWARE.** If Customer purchases a license to use development or prototyping tools of Mentor Graphics' Embedded Software Channel ("ESC"), Mentor Graphics grants to Customer a nontransferable, nonexclusive license to reproduce and distribute executable

files created using ESC compilers, including the ESC run-time libraries distributed with ESC C and C++ compiler Software that are linked into a composite program as an integral part of Customer's compiled computer program, provided that Customer distributes these files only in conjunction with Customer's compiled computer program. Mentor Graphics does NOT grant Customer any right to duplicate, incorporate or embed copies of Mentor Graphics' real-time operating systems or other embedded software products into Customer's products or applications without first signing or otherwise agreeing to a separate agreement with Mentor Graphics for such purpose.

4. **BETA CODE.**

4.1. Portions or all of certain Software may contain code for experimental testing and evaluation (which may be either alpha or beta, collectively "Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to Customer a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. Mentor Graphics may choose, at its sole discretion, not to release Beta Code commercially in any form.

4.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer's use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer's evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

4.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on Customer's feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this Subsection 4.3 shall survive termination of this Agreement.

5. **RESTRICTIONS ON USE.**

5.1. Customer may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. Customer shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. Customer shall not make Products available in any form to any person other than Customer's employees and on-site contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use Products except as permitted by this Agreement. Customer shall give Mentor Graphics written notice of any unauthorized disclosure or use of the Products as soon as Customer becomes aware of such unauthorized disclosure or use. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive any source code from Software. Log files, data files, rule files and script files generated by or for the Software (collectively "Files"), including without limitation files containing Standard Verification Rule Format ("SVRF") and Tcl Verification Format ("TVF") which are Mentor Graphics' trade secret and proprietary syntaxes for expressing process rules, constitute or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Customer may use Files containing SVRF or TVF only with Mentor Graphics products. Under no circumstances shall Customer use Products or Files or allow their use for the purpose of developing, enhancing or marketing any product that is in any way competitive with Products, or disclose to any third party the results of, or information pertaining to, any benchmark.

5.2. If any Software or portions thereof are provided in source code form, Customer will use the source code only to correct software errors and enhance or modify the Software for the authorized use. Customer shall not disclose or permit disclosure of source code, in whole or in part, including any of its methods or concepts, to anyone except Customer's employees or on-site contractors, excluding Mentor Graphics competitors, with a need to know. Customer shall not copy or compile source code in any manner except to support this authorized use.

5.3. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense, or otherwise transfer the Products, whether by operation of law or otherwise ("Attempted Transfer"), without Mentor Graphics' prior written consent and payment of Mentor Graphics' then-current applicable relocation and/or transfer fees. Any Attempted Transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and/or the licenses granted under this Agreement. The terms of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer's permitted successors in interest and assigns.

5.4. The provisions of this Section 5 shall survive the termination of this Agreement.

6. **SUPPORT SERVICES.** To the extent Customer purchases support services, Mentor Graphics will provide Customer with updates and technical support for the Products, at the Customer site(s) for which support is purchased, in accordance with Mentor Graphics' then current End-User Support Terms located at http://supportnet.mentor.com/supportterms.

7. **LIMITED WARRANTY.**

7.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Products will meet Customer's requirements or that operation of Products will be uninterrupted or error free. The

warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. Customer must notify Mentor Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Software under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes or alternate Software under a transaction involving Software re-mix. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification, improper installation or Customer is not in compliance with this Agreement. MENTOR GRAPHICS' ENTIRE LIABILITY AND CUSTOMER'S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) PRODUCTS PROVIDED AT NO CHARGE; OR (C) BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

7.2. THE WARRANTIES SET FORTH IN THIS SECTION 7 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO PRODUCTS PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

8. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT RECEIVED FROM CUSTOMER FOR THE HARDWARE, SOFTWARE LICENSE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 8 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

9. **HAZARDOUS APPLICATIONS.** CUSTOMER ACKNOWLEDGES IT IS SOLELY RESPONSIBLE FOR TESTING ITS PRODUCTS USED IN APPLICATIONS WHERE THE FAILURE OR INACCURACY OF ITS PRODUCTS MIGHT RESULT IN DEATH OR PERSONAL INJURY ("HAZARDOUS APPLICATIONS"). EXCEPT TO THE EXTENT THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF MENTOR GRAPHICS PRODUCTS IN OR FOR HAZARDOUS APPLICATIONS. THE PROVISIONS OF THIS SECTION 9 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

10. **INDEMNIFICATION.** CUSTOMER AGREES TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH THE USE OF MENTOR GRAPHICS PRODUCTS IN OR FOR HAZARDOUS APPLICATIONS. THE PROVISIONS OF THIS SECTION 10 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

11. **INFRINGEMENT.**

11.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay costs and damages finally awarded against Customer that are attributable to such action. Customer understands and agrees that as conditions to Mentor Graphics' obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

11.2. If a claim is made under Subsection 11.1 Mentor Graphics may, at its option and expense: (a) replace or modify the Product so that it becomes noninfringing; (b) procure for Customer the right to continue using the Product; or (c) require the return of the Product and refund to Customer any purchase price or license fee paid, less a reasonable allowance for use.

11.3. Mentor Graphics has no liability to Customer if the action is based upon: (a) the combination of Software or hardware with any product not furnished by Mentor Graphics; (b) the modification of the Product other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of the Product as part of an infringing process; (e) a product that Customer makes, uses, or sells; (f) any Beta Code or Product provided at no charge; (g) any software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; or (h) infringement by Customer that is deemed willful. In the case of (h), Customer shall reimburse Mentor Graphics for its reasonable attorney fees and other costs related to the action.

11.4. THIS SECTION 11 IS SUBJECT TO SECTION 8 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS, AND CUSTOMER'S SOLE AND EXCLUSIVE REMEDY, FOR DEFENSE, SETTLEMENT AND DAMAGES, WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY PRODUCT PROVIDED UNDER THIS AGREEMENT.

12. **TERMINATION AND EFFECT OF TERMINATION.**

12.1. If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized term. Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement immediately upon written notice if Customer: (a) exceeds the scope of the license or otherwise fails to comply with the licensing or confidentiality provisions of this Agreement, or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. For any other material breach of any

provision of this Agreement, Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement upon 30 days written notice if Customer fails to cure the breach within the 30 day notice period. Termination of this Agreement or any license granted hereunder will not affect Customer's obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination.

12.2. Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination, Customer shall ensure that all use of the affected Products ceases, and shall return hardware and either return to Mentor Graphics or destroy Software in Customer's possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form.

13. **EXPORT.** The Products provided hereunder are subject to regulation by local laws and United States ("U.S.") government agencies, which prohibit export, re-export or diversion of certain products, information about the products, and direct or indirect products thereof, to certain countries and certain persons. Customer agrees that it will not export or re-export Products in any manner without first obtaining all necessary approval from appropriate local and U.S. government agencies. If Customer wishes to disclose any information to Mentor Graphics that is subject to any U.S. or other applicable export restrictions, including without limitation the U.S. International Traffic in Arms Regulations (ITAR) or special controls under the Export Administration Regulations (EAR), Customer will notify Mentor Graphics personnel, in advance of each instance of disclosure, that such information is subject to such export restrictions.

14. **U.S. GOVERNMENT LICENSE RIGHTS.** Software was developed entirely at private expense. The parties agree that all Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to U.S. FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. government or a U.S. government subcontractor is subject solely to the terms and conditions set forth in this Agreement, which shall supersede any conflicting terms or conditions in any government order document, except for provisions which are contrary to applicable mandatory federal laws.

15. **THIRD PARTY BENEFICIARY.** Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, Microsoft Corporation and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.

16. **REVIEW OF LICENSE USAGE.** Customer will monitor the access to and use of Software. With prior written notice and during Customer's normal business hours, Mentor Graphics may engage an internationally recognized accounting firm to review Customer's software monitoring system and records deemed relevant by the internationally recognized accounting firm to confirm Customer's compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FlexNet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics' request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement. The provisions of this Section 16 shall survive the termination of this Agreement.

17. **CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION.** The owners of certain Mentor Graphics intellectual property licensed under this Agreement are located in Ireland and the U.S. To promote consistency around the world, disputes shall be resolved as follows: excluding conflict of laws rules, this Agreement shall be governed by and construed under the laws of the State of Oregon, U.S., if Customer is located in North or South America, and the laws of Ireland if Customer is located outside of North or South America. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the courts of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply. Notwithstanding the foregoing, all disputes in Asia arising out of or in relation to this Agreement shall be resolved by arbitration in Singapore before a single arbitrator to be appointed by the chairman of the Singapore International Arbitration Centre ("SIAC") to be conducted in the English language, in accordance with the Arbitration Rules of the SIAC in effect at the time of the dispute, which rules are deemed to be incorporated by reference in this section. Nothing in this section shall restrict Mentor Graphics' right to bring an action (including for example a motion for injunctive relief) against Customer in the jurisdiction where Customer's place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

18. **SEVERABILITY.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

19. **MISCELLANEOUS.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements. Some Software may contain code distributed under a third party license agreement that may provide additional rights to Customer. Please see the applicable Software documentation for details. This Agreement may only be modified in writing, signed by an authorized representative of each party. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

Rev. 140201, Part No. 258976