



A Siemens Business

Questa® SIM Command Reference Manual

Including Support for Questa SV/AFV

Software Version 10.7c

Document Revision 3.6

**© 1991-2018 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.

Note - Viewing PDF files within a web browser causes some links not to function (see [MG595892](#)).
Use HTML for full navigation.

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: 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.

End-User License Agreement: You can print a copy of the End-User License Agreement from: mentor.com/eula.

Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: mentor.com
Support Center: support.mentor.com

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

Revision History

Revision	Changes	Status/ Date
3.6	<p>Modifications to improve the readability and comprehension of the content. Approved by Farshad Dailami.</p> <p>All technical enhancements, changes, and fixes listed in the Release Notes are reflected in this document.</p> <p>Approved by Tim Peeke.</p>	Released August 2018
3.5	<p>Modifications to improve the readability and comprehension of the content. Approved by Farshad Dailami.</p> <p>All technical enhancements, changes, and fixes listed in the Release Notes are reflected in this document.</p> <p>Approved by Tim Peeke.</p>	Released June 2018
3.4	<p>Modifications to improve the readability and comprehension of the content. Approved by Farshad Dailami.</p> <p>All technical enhancements, changes, and fixes listed in the Release Notes are reflected in this document.</p> <p>Approved by Tim Peeke.</p>	Released March 2018
3.3	<p>Modifications to improve the readability and comprehension of the content. Approved by Farshad Dailami.</p> <p>All technical enhancements, changes, and fixes listed in the Release Notes are reflected in this document.</p> <p>Approved by Tim Peeke.</p>	Released December 2017

Author: In-house procedures and working practices require multiple authors for documents. All associated authors for each topic within this document are tracked within the Mentor Graphics Technical Publication's source. For specific topic authors, contact Mentor Graphics Technical Publication department.

Revision History: Released documents maintain a revision history of up to four revisions. For earlier revision history, refer to earlier releases of documentation which are available on Support Center (<http://support.mentor.com>).

Table of Contents

Revision History

Chapter 1		
Syntax and Conventions		19
Documentation Conventions		19
File and Directory Pathnames		20
Design Object Names		21
Object Name Syntax		21
Tcl Syntax and Specification of Array Bits and Slices		22
SystemC Class, Structure, and Union Member Specification		23
SystemVerilog Scope Resolution Operator		24
Specifying Names		26
Environment Variables and Pathnames		28
Name Case Sensitivity		28
Extended Identifiers		28
Wildcard Characters		30
Supported Commands		30
Using the WildcardFilter Preference Variable		31
Simulator Variables		35
Simulation Time Units		35
Optionsets		35
Argument Files		37
Command Shortcuts		38
Command History Shortcuts		39
Numbering Conventions		40
VHDL Numbering Conventions		40
Verilog Numbering Conventions		41
GUI_expression_format		42
Expression Typing		42
Expression Syntax		43
Signal and Subelement Naming Conventions		49
Grouping and Precedence		49
Concatenation of Signals or Subelements		49
Record Field Members		51
Searching for Binary Signal Values in the GUI		52
Chapter 2		
Commands (A - M)		55
.main clear		72
.abort		73
.add atv		74
.add button		75

Table of Contents

add dataflow	77
add list	79
add memory	84
add message	86
add schematic	88
add testbrowser	90
add watch	91
add wave	92
add_cmdhelp	100
add_menu	102
add_menucb	104
add_menuitem	106
add_separator	108
add_submenu	109
alias	111
archive load	112
archive write	113
assertion action	114
assertion active	117
assertion count	119
assertion enable	121
assertion fail	125
assertion pass	128
assertion profile	130
assertion report	132
atv log	136
batch_mode	138
bd	139
bookmark add wave	141
bookmark delete wave	143
bookmark goto wave	144
bookmark list wave	145
bp	146
call	153
capstats	157
cd	158
cdbg	159
change	162
change_menu_cmd	165
check contention add	166
check contention config	168
check contention off	169
check float add	170
check float config	172
check float off	173
check stable off	174
check stable on	175
checkpoint	177
classinfo ancestry	180

Table of Contents

classinfo descriptive	181
classinfo find	183
classinfo implements	185
classinfo instances	187
classinfo interfaces	190
classinfo isa	192
classinfo report	193
classinfo stats	195
classinfo trace	197
classinfo types	199
compare add	201
compare annotate	207
compare clock	209
compare configure	211
compare continue	213
compare delete	214
compare end	215
compare info	216
compare list	218
compare options	219
compare reload	224
compare reset	225
compare run	226
compare savediffs	227
compare saverules	228
compare see	229
compare start	231
compare stop	233
compare update	234
configure	235
context	241
coverage analyze	245
coverage attribute	255
coverage clear	260
coverage create	263
coverage edit	266
coverage exclude	272
coverage goal	283
coverage loadtestassoc	287
coverage open	288
coverage ranktest	289
coverage report	297
coverage save	311
coverage tag	315
coverage testnames	322
coverage unlinked	323
coverage weight	327
dataset alias	331
dataset clear	332

dataset close	334
dataset config	335
dataset current	337
dataset info	338
dataset list	339
dataset open	340
dataset rename	341
dataset restart	342
dataset save	343
dataset snapshot	344
delete	347
describe	348
disablebp	350
disable_menu	351
disable_menuitem	352
do	353
down	355
drivers	357
dumplog64	359
echo	360
edit	361
enablebp	362
enable_menu	363
enable_menuitem	364
encoding	365
environment	366
examine	368
exit	376
fcover configure	377
find	380
find connections	386
find drivers	387
find infiles	392
find insource	393
force	395
formatTime	401
fsm list	402
fsm properties	403
fsm view	404
gc configure	405
gc run	407
gdb dir	408
getactivecursortime	409
getactivemarkertime	410
help	411
history	412
jobspray	413
layout	415
lecho	417

Table of Contents

left	418
log	421
lshift	425
lsublist	426
mem compare	427
mem display	428
mem list	431
mem load	432
mem save	436
mem search	439
msgZeroMessageCounts	442
Chapter 3	
Commands (N - Z)	443
next.	465
noforce	466
nolog	467
notepad.	469
noview	470
nowhen.	471
onbreak.	472
onElabError	474
onerror	475
onfinish	477
pa autotestplan	478
pa msg	479
pa report	485
pause	486
pop	487
power add	488
power off	490
power on	493
power report	496
power reset	499
precision	501
printenv	502
process report	503
profile clear	504
profile configure	505
profile interval	507
profile off	508
profile on	510
profile open	512
profile option	513
profile reload	515
profile report	517
profile save	521
profile status	523

profile summary	524
project	527
property list	530
property wave	532
push	534
pwd	535
questasim	536
quietly	537
quit	538
qverilog	539
radix	543
radix define	546
radix delete	550
radix list	551
radix names	552
radix signal	553
readers	554
report	555
restart	558
restore	560
resume	561
right	562
run	565
runStatus	568
sccom	570
scgenmod	581
sdfcom	585
search	587
searchlog	590
see	593
seetime	594
setenv	595
shift	596
show	597
simstats	598
simstatslist	601
stack down	603
stack frame	604
stack level	605
stack tb	606
stack up	607
status	608
step	609
stop	611
suppress	612
sv_reseed	614
tb	615
tcheck_set	616
tcheck_status	621

Table of Contents

Time	625
toggle add	629
toggle disable	633
toggle enable	635
toggle report	636
toggle reset	638
tr color	639
tr order	642
tr uid	644
transcribe	646
transcript	647
transcript file	648
transcript path	650
transcript sizelimit	651
transcript wrapcolumn	652
transcript wrapmode	653
transcript wrapwscolumn	654
triage dbfile	655
triage dbinsert	661
triage dbrefresh	663
triage passfail	664
triage query	667
triage report	670
triage view	673
tssi2mti	675
typespec	676
ui_VVMode	677
unsetenv	679
up	680
uvm call	683
uvm configtracing	686
uvm displayobjections	688
uvm findregisters	691
uvm findsequences	694
uvm handle	696
uvm mapmode	698
uvm printconfig	700
uvm printfactory	703
uvm printstreams	706
uvm printtopology	708
uvm setverbosity	710
uvm simpAth	712
uvm traceobjections	714
uvm uvmpath	716
vcd add	718
vcd checkpoint	721
vcd comment	722
vcd dumports	723
vcd dumportsall	726

vcd dumpsupportsflush	727
vcd dumpportslimit	728
vcd dumpportsoff	730
vcd dumpportson	731
vcd file	732
vcd files	734
vcd flush	737
vcd limit	739
vcd off	741
vcd on	742
vcd2wlf	744
vcom	746
vcover attribute	772
vcover diff	777
vcover dump	782
vcover history	785
vcover merge	788
vcover parallelmerge	798
vcover ranktest	803
vcover remove	814
vcover report	816
vcover stats	838
vcover testnames	843
vdel	847
vdir	849
vencrypt	852
verror	859
vgencomp	861
vhencrypt	863
view	866
virtual count	870
virtual define	871
virtual delete	872
virtual describe	873
virtual expand	874
virtual function	875
virtual hide	879
virtual log	880
virtual nohide	882
virtual nolog	883
virtual region	885
virtual save	886
virtual show	887
virtual signal	888
virtual type	892
vlib	894
vlog	897
vmake	936
vmap	939

Table of Contents

vopt	941
vsim	1009
vsim<info>	1079
vsim_break	1080
vsource	1081
wave	1082
wave create	1086
wave edit	1092
wave export	1095
wave import	1097
wave modify	1098
wave sort	1103
when	1104
where	1112
wlf2log	1113
wlf2vcd	1116
wlfman	1117
wlfrecover	1124
write cell_report	1125
write format	1126
write list	1129
write preferences	1130
write report	1131
write timing	1135
write transcript	1137
write tssi	1138
write wave	1140
xml2ucdb	1142
xprop assertlimit	1149
xprop disable	1150
xprop enable	1151

Index

End-User License Agreement

List of Figures

Figure 2-1. Graphical representation of tolLead and tolTrail - compare add	204
Figure 2-2. Graphical representation of tolLead and tolTrail	221
Figure 2-3. drivers Command Results in Transcript	357
Figure 2-4. find infiles Example	392
Figure 2-5. find insource Example	394
Figure 3-1. readers Command Results in Transcript	554
Figure 3-2. UVM Sequences	695

List of Tables

Table 1-1. Conventions for Command Syntax	19
Table 1-2. Examples of Object Names	27
Table 1-3. Wildcard Characters in HDL Object Names	31
Table 1-4. WildcardFilter Arguments	33
Table 1-5. WildcardFilter Argument Groups	34
Table 1-6. Keyboard Shortcuts for Command History	39
Table 1-7. VHDL Number Conventions: Style 1	40
Table 1-8. VHDL Number Conventions: Style 2	40
Table 1-9. Verilog Number Conventions	41
Table 1-10. Constants Supported for GUI Expressions	43
Table 1-11. Array Constants Supported for GUI Expressions	44
Table 1-12. Variables Supported for GUI Expressions	44
Table 1-13. Array Variables Supported for GUI Expressions	45
Table 1-14. Operators Supported for GUI Expressions	45
Table 1-15. Precedence of GUI Expression Operators	47
Table 1-16. Casting Conversions Supported for GUI Expressions	48
Table 1-17. VHDL Logic Values Used in GUI Search	52
Table 1-18. Verilog Logic Values Used in GUI Search	53
Table 2-1. Supported Commands	55
Table 2-2. Message Viewer Categories	86
Table 2-3. Comparing Reference Objects to Test Objects	201
Table 2-4. Order and Type of Ranked Tests	289
Table 2-5. Radix flag Arguments to the Examine Command	372
Table 3-1. Supported Commands N-Z	443
Table 3-2. profile summary Header Rules for Return	525
Table 3-3. runStatus Command States	568
Table 3-4. runStatus -full Command Information	568
Table 3-5. Field Arguments for Window Searches	588
Table 3-6. Output Fields for tcheck_status Command	622
Table 3-7. Alias for Tables	667
Table 3-8. Warning Message Categories for vcom -nowarn	763
Table 3-9. Order and Type of Ranked Tests	804
Table 3-10. Design Unit Properties	850
Table 3-11. Warning Message Categories for vlog -nowarn	921
Table 3-12. Warning Message Categories for vopt -nowarn	975
Table 3-13. Wave Window Commands for Cursor	1082
Table 3-14. Wave Window Commands for Expanded Time Display	1083
Table 3-15. Wave Window Commands for Controlling Display	1084
Table 3-16. Wave Window Commands for Zooming	1084
Table 3-17. when_condition_expression Operators	1107

Chapter 1

Syntax and Conventions

This chapter describes the typographical conventions used in this manual to define Questa SIM command syntax.

Documentation Conventions	19
File and Directory Pathnames	20
Design Object Names	21
Wildcard Characters	30
Simulator Variables	35
Simulation Time Units	35
Optionsets	35
Argument Files	37
Command Shortcuts	38
Command History Shortcuts	39
Numbering Conventions	40
GUI_expression_format	42

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

Table 1-1. Conventions for Command Syntax (cont.)

Syntax notation	Description
...	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 (#), which you can use to add 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

 Command examples do not show either the prompt at the beginning of a line nor the <Enter> keystroke at the end of a line.

File and Directory Pathnames

Several Questa SIM commands have arguments that specify file or directory locations (pathnames). For example, the **-y** argument to the vlog command 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 quotation marks. 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	21
Tcl Syntax and Specification of Array Bits and Slices	22
SystemC Class, Structure, and Union Member Specification	23
SystemVerilog Scope Resolution Operator	24
Specifying Names	26
Environment Variables and Pathnames	28
Name Case Sensitivity	28
Extended Identifiers	28

Object Name Syntax

To specify object names in Questa SIM, you use the following syntax:

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

where

- **dataset_name** — 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 “[Recording Simulation Results With Datasets](#)” in the User’s Manual for more information.
- **datasetSeparator** — The character used to terminate the dataset name. The default is colon (:), although you can specify a different character (except for a backslash (\)) as the dataset separator by using the DatasetSeparator variable in the *modelsim.ini* file. (Refer to [DatasetSeparator](#) in the User’s Manual.) This character must be different than the pathSeparator character.
- **pathSeparator** — The character used to separate hierarchical object names. Normally, a backslash (\) is used for VHDL and a period (.) is used for Verilog, although you can specify a different character (except for a backslash (\)) by using the PathSeparator

variable in the *modelsim.ini* file. (Refer to [PathSeparator](#) in the User's Manual.) 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** — 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** — The name of an object in a design.
- **elementSelection** — 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 based on the Tcl scripting language, you must surround objects and signals with 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 noted, square brackets 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 do not want this type of behavior when specifying an array slice, so you would use brace escape characters, as follows:

```
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)}
```

Refer to [Tcl Command Syntax](#) in the User's Manual for more information.

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>
```

However, in Questa SIM you *must* include the name of the base class according to the following:

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

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

The following example shows 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 examples of the `examine` command 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

Questa SIM distinguishes 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 (such as *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 (such as *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 (for example, */top/u1/clk*). There is a special case of a fully-rooted name where the top-level design unit name can be unspecified (such as */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 (such as *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
<i>clk</i>	specifies the object <i>clk</i> in the current context
<i>/top/clk</i>	specifies the object <i>clk</i> in the top-level design unit.
<i>/top/block1/u2/clk</i>	specifies the object <i>clk</i> , two levels down from the top-level design unit
<i>block1/u2/clk</i>	specifies the object <i>clk</i> , two levels down from the current context
<i>array_sig[4]</i>	specifies an index of an array object

Table 1-2. Examples of Object Names (cont.)

Object Name	Description
{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](#) in the User's Manual 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	30
Using the WildcardFilter Preference Variable	31

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
- dumports
- 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

[WildcardSizeThreshold](#) and [WildcardSizeThresholdVerbose](#) in the User's Manual 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 <i>only</i> 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. (Refer to [WildcardFilter](#) in the User's Manual for more information.) 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](#) in the User's Manual).
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.
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

Table 1-4. WildcardFilter Arguments (cont.)

Argument	Description
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
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

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

Group Argument	Specific arguments included
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.

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. Refer to [UserTimeUnit](#) in the User's Manual for more information.

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 = +includer+${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 +includer+/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 conventions describe 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
```

- Quotation marks (“ ”) — 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 substitutes  
// 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 a backslash (\), which is 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 can abbreviate command syntax, but the minimum number of characters required to run a command are those that make it unique. This means the addition of new commands may prevent an older shortcut from working. 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.
- You can enter multiple commands 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
```

Although it seems as if the **simstats** results should display in the Transcript window, they do 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 prompt	left-click 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	40
Verilog Numbering Conventions	41

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 hyphen (-) can also 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 as a negative sign, be sure to enclose the number in quotation marks. For instance, you would type "-0110--" as opposed to -0110--. If you do not include the quotation marks, 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

Table 1-8. VHDL Number Conventions: Style 2 (cont.)

Element	Description
"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: ' <code>b</code> ' or ' <code>B</code> ', octal: ' <code>o</code> ' or ' <code>O</code> ', decimal: ' <code>d</code> ' or ' <code>D</code> ', hex: ' <code>h</code> ' or ' <code>H</code> '; optional
value	specifies digits in the appropriate base with optional underscore separators; default is decimal; required

A hyphen (-) can also 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 do not include the double quotes, Questa SIM will read the '-' as a negative sign. For example:

<code>'b11111110</code>	<code>8'b11111110</code>
<code>'Hffca23</code>	<code>21'H1fca23</code>
<code>-23749</code>	

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

Expression Typing	42
Expression Syntax	43
Signal and Subelement Naming Conventions	49
Grouping and Precedence	49
Concatenation of Signals or Subelements	49
Record Field Members	51
Searching for Binary Signal Values in the GUI	52

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>, and so on.

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]+

Table 1-10. Constants Supported for GUI Expressions (cont.)

Type	Values
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...., 0B.... 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: <ul style="list-style-type: none"> • 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 (bool, int, and so on)

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	<ul style="list-style-type: none"> VHDL signals of type bit_vector or std_logic_vector Verilog register Verilog net array SystemC primitive channels of type vector (sc_bv, sc_int, and so on) 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 (for example, #-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

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

Operator	Description	Kind
-	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
==	equal	boolean
====	exact equal ¹	boolean
!==	exact not equal	boolean
>	greater than	boolean
>=	greater than or equal	boolean
<	less than	boolean

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

Operator	Description	Kind
<code><=</code>	less than or equal	boolean
<code>!=, /=</code>	not equal	boolean
<code>&<vector_expr></code>	AND reduction	reduction
<code> <vector_expr></code>	OR reduction	reduction
<code>^<vector_expr></code>	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
<code>&, , ^</code>	unary
<code>!, not, NOT, ~</code>	boolean
<code>/, mod, MOD, *, rem, REM</code>	arithmetic
nand, NAND, nor, NOR	bitwise logical
and, AND	bitwise logical
xor, XOR, xnor, XNOR	bitwise logical
or, OR	bitwise logical
<code>+, -</code>	arithmetic
<code>&</code>	concat
rol, ROL, ror, ROR, sla, SLA, sll, SLL, sra, SRA, srl, SRL	bitwise logical
<code>>, >=, <, <=</code>	boolean
<code>==, ===, !=, /=</code>	boolean
<code>&&</code>	boolean
<code> </code>	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 liberal use of parentheses is recommended.

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, leaving full arrays as elements in the result is not supported. (Please contact Mentor Graphics 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, Questa SIM will create the concatenation 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_descending) <concatenationExpr>
```

The **concat_descending** 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_wildAscending) <concatenationExpr>
```

The **concat_sort_wildAscending** 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.

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

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

```
(concatAscending) ((concatSortWildAscending) &{ "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:

```
vhdsig.field1
vhdsig.field1.subfield1
vhdsig.(5).field3
vhdsig.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_bignum.

Chapter 2

Commands (A - M)

This chapter describes Questa SIM commands, listed alphabetically from A through M, 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 [General 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	This command clears the Main window Transcript window.	N	N
abort	This command halts the execution of a DO file interrupted by a breakpoint or error.	Y	Y
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.	N	N
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.	N	N
add dataflow	This command adds the specified process, signal, net, or register to the Dataflow window. Wildcards are allowed.	N	N
add list	This command adds the following objects and their values to the List window:	Y	Y
add log	also known as the log command; see “ log ” on page 421	Y	Y
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.	N	N

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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 .wlffiles. Messages are displayed in the Message Viewer window in the GUI. Refer to “Message Viewer Window for information.	Y	Y
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.	N	N
add testbrowser	This command adds .ucdb file(s) to the test management browser.	N	N
add watch	This command adds signals and variables to the Watch window in the Main window.	N	N
add wave	This command adds the following objects to the Wave window:	Y	Y
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.	N	Y
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.	N	N
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.	N	N
add_menuitem	This command creates a menu item within the specified menu of the specified window. May be used within a submenu.	N	N
add_separator	This command adds a separator as the next item in the specified menu path in the specified window.	N	N
add_submenu	This command creates a cascading submenu within the specified menu path of the specified window. May be used within a submenu.	N	N

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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.	Y	Y
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.	N	N
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.	N	N
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.	Y	Y
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.	Y	Y
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.	Y	Y
assertion enable	This command enables and disables assertions and cover directives.	Y	Y
assertion fail	This command configures simulator behavior in response to a SystemVerilog or PSL assertion failure.	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
assertion pass	This command configures simulator behavior in response to a SystemVerilog or PSL assertion pass.	Y	Y
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.	Y	Y
assertion report	This command generates a status report for each SystemVerilog or PSL assertion matching the path specification.	Y	Y
atv log	This command enables or disables assertion thread viewing (ATV) for the specified assertion. Multiple assertions may be specified by their pathnames.	Y	Y
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.	Y	Y
bd	This command deletes a breakpoint. You can delete multiple breakpoints by specifying separate information groupings on the same command line.	Y	Y
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.	N	N
bookmark delete wave	This command deletes bookmarks from the specified Wave window.	N	N
bookmark goto wave	This command zooms and scrolls a Wave window using the specified bookmark.	N	N
bookmark list wave	This command displays a list of available bookmarks in the Transcript window.	N	N
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.	Y	Y
call	This command calls the following types of functions/tasks.	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
<code>cd</code>	This command changes the Questa SIM local directory to the specified directory.	Y	Y
<code>cdbg</code>	This command provides command-line equivalents of the menu options that are available for C Debug.	N	Y
<code>change</code>	This command modifies the value of a: VHDL constant, generic, or variable; Verilog register or variable; or C variable if running C Debug.	Y	Y
<code>change_menu_cmd</code>	This command changes the command to be executed for a specified menu item label, in the specified menu, in the specified window.	N	N
<code>check contention add</code>	This command enables contention checking for the specified nodes.	Y	Y
<code>check contention config</code>	This command allows you to write checking messages to a file. By default, any messages display on your screen.	Y	Y
<code>check contention off</code>	This command disables contention checking for the specified nodes.	Y	Y
<code>check float add</code>	This command enables float checking for the specified nodes.	Y	Y
<code>check float config</code>	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.	Y	Y
<code>check float off</code>	This command disables float checking for the specified nodes.	Y	Y
<code>check stable off</code>	This command disables stability checking.	Y	Y
<code>check stable on</code>	This command enables stability checking on the entire design.	Y	Y
<code>checkpoint</code>	This command saves the state of your simulation, including:	Y	Y
<code>classinfo ancestry</code>	This command returns class inheritance hierarchy for a named class type.	Y	Y
<code>classinfo descriptive</code>	This command returns the descriptive class name for the specified authoritative class name.	Y	Y
<code>classinfo find</code>	This command reports on the current state of a specified class instance, whether it exists, has not yet been created, or has been destroyed.	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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.	Y	Y
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.	Y	Y
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.	Y	Y
classinfo isa	This command returns to the transcript a list of all classes extended from the specified class type.	Y	Y
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.	Y	Y
classinfo stats	This command prints statistics about the total number of class types and total, peak, and current class instance counts during the simulation.	Y	Y
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.	Y	Y
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.	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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.	N	Y
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.	N	Y
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.	N	Y
compare configure	This command modifies options for compare signals and regions. The modified options are applied to all objects in the specified compare path.	N	Y
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.	N	Y
compare delete	This command deletes a comparison object from the currently open comparison.	N	Y
compare end	This command closes the active comparison without saving any information.	N	Y
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.	N	Y
compare list	Displays in the Transcript window a list of all the compare add commands currently in effect.	N	Y
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.	N	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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.	N	Y
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.	N	Y
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.	N	Y
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.	N	Y
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.	N	Y
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.	N	Y
compare start	This command begins a new dataset comparison. The datasets that you will be comparing must already be open.	N	Y
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.	N	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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.	N	Y
configure	The configure command invokes the List or Wave widget configure command for the current default List or Wave window.	N	Y
context	This command provides several operations on a context's name. The option you specify determines the operation.	Y	Y
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.	N	Y
coverage attribute	The coverage attribute command is used to display or set attributes in the currently loaded database on the following types of attributes:	Y	Y
coverage clear	The coverage clear command clears specified types of coverage data from the coverage database.	Y	Y
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.	N	Y
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.	N	Y
coverage exclude	The coverage exclude command allows you to exclude the following from coverage statistics:	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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:	N	Y
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).	Y	Y
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.	N	Y
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.	N	Y
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.	Y	Y
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.	Y	Y
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.	N	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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.	N	Y
coverage unlinked	The coverage unlinked command is used as a diagnostic tool to report on a number of items.	N	Y
coverage weight	The coverage weight command is used to:	N	Y
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.	N	Y
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.	N	Y
dataset close	This command closes an active dataset. To open a dataset, use the dataset open command.	N	Y
dataset config	This command configures WLF parameters for an open dataset and all aliases mapped to that dataset.	N	Y
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.	N	Y
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.	N	Y
dataset list	This command lists all active datasets.	N	Y
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.	N	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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.	N	Y
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.	N	Y
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.	N	Y
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.	N	Y
delete	This command removes objects from either the List or Wave window. Arguments to this command are order dependent.	N	Y
describe	This command displays information about simulation objects and design regions in the Transcript window.	Y	Y
disablebp	This command turns off breakpoints and when commands. To turn on breakpoints or when commands again, use the enablebp command.	Y	Y
disable_menu	This command disables the specified menu within the specified window.	N	N
disable_menuitem	This command disables a specified menu item within the specified menu path of the specified window.	N	N
do	This command executes the commands contained in a DO file.	Y	Y
down	This command searches for object transitions or values in the specified List window.	N	Y
drivers	This command displays the names and strength of all drivers of the specified object.	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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.		
echo	This command displays a specified message in the Transcript window.	Y	Y
edit	This command invokes the editor specified by the EDITOR environment variable. By default, the specified filename will open in the Source window.	N	N
enablebp	This command turns on breakpoints and when commands that were previously disabled.	Y	Y
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.	N	N
enable_menuitem	This command enables a previously disabled menu item.	N	N
encoding	This command translates between the 16-bit Unicode characters used in Tcl strings and a named encoding, such as Shift-JIS.	Y	Y
environment	This command has two forms, environment and env. It allows you to display or change the current dataset and region/signal environment.	Y	Y
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.	Y	Y
exit	This command exits the simulator and the ModelSim application.	Y	Y
fcover configure	This command enables, disables, and sets coverage targets for SystemVerilog and PSL cover directives.	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
find	This command locates objects by type and name. Arguments to the command are grouped by object type. The “find nets”, “find signals”, and “find instances” commands are supported in -batch mode.	N	Y
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.	N	Y
find drivers	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.	N	Y
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.	Y	Y
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.	Y	Y
force	This command allows you to apply stimulus interactively to VHDL signals, Verilog nets and registers.	Y	Y
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.	Y	Y
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.	N	N

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
fsm properties	This command returns information about the specified finite state machine. This information matches that found in the FSM Properties dialog box.	N	N
fsm view	This command displays the specified finite state machine in the FSM List window.	N	N
gc configure	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.	N	Y
gc run	This command runs the SystemVerilog garbage collector.	N	Y
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.	N	Y
getactivecursortime	This command gets the time of the active cursor in the Wave window and returns the time value.	N	N
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.	N	N
help	This command displays in the Transcript window a brief description and syntax for the specified command.	N	Y
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).	Y	Y
jobsy	This command controls JobSpy, a tool for monitoring and controlling batch simulations and simulation farms. This command provides additional information with the -help switch.	N	Y
layout	This command allows you to perform a number of editing operations on custom GUI layouts, such as loading, saving, maximizing, and deleting.	N	N

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
lecho	This command takes one or more Tcl lists as arguments and pretty-prints them to the Transcript window.	Y	Y
left	This command searches left (previous) for signal transitions or values in the specified Wave window.	N	N
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.	Y	Y
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.	Y	Y
l sublist	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.	Y	Y
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.	Y	Y
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.	Y	Y
mem list	This command displays a flattened list of all memory instances in the current or specified context after a design has been elaborated.	Y	Y

Table 2-1. Supported Commands (cont.)

Command name	Action	-batch	-c
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.	Y	Y
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.	Y	Y
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.	Y	Y

.main clear

Clears the Main window Transcript window.

Note

 Selecting **Edit > Clear** when the Transcript window is active produces the same results.

Syntax

.main clear

Arguments

None

Related Topics

[Main Window \[Questa SIM GUI Reference Manual\]](#)

[Transcript Window \[Questa SIM GUI Reference Manual\]](#)

abort

Halts the execution of a DO file interrupted by a breakpoint or error.

When DO files are nested, you can choose to abort the last DO file script only, to abort a specified number of nesting levels, or to 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.

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. Arguments to this command are order-dependent.

Syntax

`add atv <pathname> <time>`

Arguments

- `<pathname>`
(required) Specifies the path of an assert or cover directive. Must be the first argument.
- `<time>`
(required) Specifies the start time of an evaluation attempt of the assert or cover directive instance. Must be the second argument.

Related Topics

[atv log](#)

[Viewing Assertion Threads in the ATV Window \[Questa SIM User's Manual\]](#)

add button

Adds a user-defined button to the Main window button bar. New buttons appear on the right side of the Standard toolbar.

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 the first argument.
- <cmd>[; ...]
(required) Specifies the command(s) to execute when the button is clicked. Separate multiple commands with a semicolon (;). Must be the second argument.
Commands that contain any spaces or non-alphanumeric characters must be enclosed in braces ({}).
To echo the command and display the return value in the Transcript window, prefix the command with the [transcribe](#) command. The transcribe command also echos 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) 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).

Enter multiple <option><value> Tk button options 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 from [Help > Tel Man Pages](#). 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
```

Description

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 previously added button, you can use the **destroy** Tcl command with the button's path name as an argument. For example:

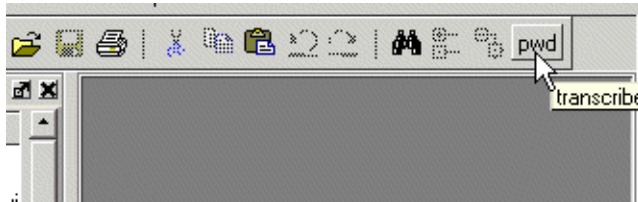
```
destroy .dockbar.tbf0.standard.tb.button_49
```

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

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. To do this, you need to 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
```

add dataflow

Adds the specified process, signal, net, or register to the Dataflow window. Wildcards are allowed.

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. Must be specified as the first argument to the add dataflow command. Wildcards are allowed. Multiple objects are specified as a space separated list, Refer to the section “[Wildcard Characters](#)” on page 30 for wildcard usage as it pertains to the add commands.
- -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 \[Questa SIM User's Manual\]](#)

[Dataflow Window \[Questa SIM GUI Reference Manual\]](#)

add list

Adds objects and their values to the List window.

Syntax

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

Description

Use add list to display the following types of objects and their values in 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

- <object> ...

(required, if <object_name>{sig ...} is not specified.) Specifies the name of the object to be listed. When you use this argument, you must specify it as the first argument to the add list command. Enter multiple objects as a space separated list. Wildcards are allowed. Refer to the section “[Wildcard Characters](#)” for wildcard usage as it pertains to the add commands.

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
```

- <object_name> {sig ...}

(required, if <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. When you use this argument, you must specify it as the first argument to the add list command. You must enclose the full argument set in braces ({}).

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 want to specify. Valid filters, <f>, are exactly the same set of words that you can apply to the WildcardFilter. The order filters are applied during a command is WildcardFilter first, then any user specified filters. The **-filter** values are added to the filter, the **-nofilter** values are removed from the filter. The filters 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. (Refer to [DefaultRadix](#) in the User's Manual.)

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 variable in the *modelsim.ini* file) for the current simulation only.

- **-radixenumnumeric**
This option overrides the global setting of the default radix (the variable in the modelsim.ini file).
- **-radixenumsymbolic**
(optional) Reverses the action of **-radixenumnumeric** and sets the global setting of the default radix (the 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](#)

add memory

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.

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. (Refer to [DefaultRadix](#) in the User’s Manual.) 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. Must be specified as the final argument to the add memory command. Multiple memories are specified as a space separated list.

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 \[Questa SIM GUI Reference Manual\]](#)

add message

Used within a DO file or script to specify a user-defined runtime message to be sent to the transcript and .wlf files. Messages are displayed in the Message Viewer window in the GUI. Refer to the *GUI Reference Manual* for more information on this window.

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 Message Viewer window Time column lists the time specified 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 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) Displays the message 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 \[Questa SIM User's Manual\]](#)
[msgmode \[Questa SIM User's Manual\]](#)
[Message Viewer Window \[Questa SIM GUI Reference Manual\]](#)

add schematic

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

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. Must be specified as the first argument to the add schematic command. Specify multiple objects as a space separated list. Wildcards are allowed. Refer to the section “[Wildcard Characters](#)” on page 30 for wildcard usage as it pertains to the add commands.
- -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 open, 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

[Automatically Tracing All Paths Between Two Nets \[Questa SIM User's Manual\]](#)

[Schematic Window \[Questa SIM User's Manual\]](#)

add testbrowser

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 add. Specify multiple filenames as a space separated list.

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

Related Topics

[“Verification Management Browser Window” \[Questa SIM GUI Reference Manual\]](#)

add watch

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.

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 the first argument to the add watch command.

Wildcard characters are allowed. (Note that the *WildcardFilter* Tcl preference variable identifies the types to ignore when matching objects with wildcard patterns.) Wildcard expansion is limited to 150 items. If you exceed this limit, a dialog box will ask you to accept the limit or cancel the operation.

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 with the [radix](#) command. You can change the default radix permanently by editing the DefaultRadix variable in the *modelsim.ini* file. (Refer to [DefaultRadix](#) in the User’s Manual.)

- -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 \[Questa SIM GUI Reference Manual\]](#)

[DefaultRadix \[Questa SIM User's Manual\]](#)

add wave

Adds objects to the Wave window.

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] [-mvcrecccomplete] [-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 ...}}]
```

Description

Use add wave to display the following types of objects in 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 displays all objects in the selected region with names matching the object name specification.

Refer to “[Wildcard Characters](#)” on page 30 for wildcard usage as it pertains to the add commands.

Arguments to this command are order dependent, as described in the argument descriptions.

Arguments

- **-allowconstants**
(optional) *For use with wildcard searches.* Specifies to add constants matching the wildcard search 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
`-depths 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. Enter multiple objects 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 quotation marks ("') 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 want to specify. Valid filters, `<f>`, are the exact set of words that apply to the WildcardFilter. The WildcardFilter is used first during a command, followed by 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:

<code>-format <type></code>	<code>-<format></code>	Display Format
<code>-format literal</code>	<code>-literal</code>	Literal waveforms are displayed as a box containing the object value.
<code>-format logic</code>	<code>-logic</code>	Logic signals may be U, X, 0, 1, Z, W, L, H, or '-'.
<code>-format analog-step</code>	<code>-analog-step</code>	Analog-step changes to the new time before plotting the new Y.
<code>-format analog-interpolated</code>	<code>-analog-interpolated</code>	Analog-interpolated draws a diagonal line.
<code>-format analog-backstep</code>	<code>-analog-backstep</code>	Analog-backstep plots the new Y before moving to the new time.
<code>-format event</code>	<code>-event</code>	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](#)” in the User’s Manual).

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 quotation marks ("") or braces ({}) if it contains any spaces.

`<sig_name> ...` — Specifies the signals to add to the group. Enter multiple signals 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 display 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 display 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.
- **-mvcrecccomplete**
(optional) Enables the logging of all Questa Verification IP transaction instances that 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 execute 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 you specify the value 3, then only three elements display 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 you do not specify a radix 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. (Refer to [DefaultRadix](#) variable in the User's Manual.)

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. You can use the **-radix <type>** switch 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 `variable` in the *modelsim.ini* file) for the current simulation only.

- **-radixenumnumeric**

(optional) Causes Verilog and SystemC enums to display 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 include in the Wave window. Must be specified as the final argument to the add wave command. Wildcard characters are allowed. Enter multiple objects as a space separated list. Note that the *WildcardFilter* Tcl preference variable identifies types to ignore when matching objects with wildcard patterns.

You can add variables by preceding them with 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 the final argument to the add wave command.
 sig — A space-separated list, enclosed in braces ({ }), of the signals 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](#)

[Wave Window \[Questa SIM GUI Reference Manual\]](#)

add_cmdhelp

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

Note

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

Syntax

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

Arguments

- {<command_name>}
(required) Specifies the command name to enter as an argument to the help command. Must be enclosed in braces ({ }). The command_name cannot be the same as an 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 does not 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

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.

You can change color and other Tk properties of the menu, after creation, with the Tk menu widget configure command.

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 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 corresponds to the place of a letter in the menu name that is to be used as the 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.
- -hide_menubutton
(optional) Specifies not to display the new menu. 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. Also adds a checkbox that controls the value of myglobalvar (.signals:one).

```

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 DO file found at <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 \[Questa SIM GUI Reference Manual\]](#)

add_menucb

Creates a checkbox within the specified menu of the specified window. A checkbox is a small box with a label. Clicking on the box toggles the state from on to off, or off to on.

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>. If any operation changes the global variable, the state of the checkbox also changes. Returns nothing.

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 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 the second argument to the add_menucb command.
- <text>
(required) Specifies the text to display next to the checkbox. Must be the third argument to the add_menucb command.
- -indicatoron {0 | 1}
(optional) Specifies whether or not to display the status indicator.
 - 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 in the [add_menu](#) example.

Related Topics

[add_menu](#)

[add_separator](#)

[add_submenu](#)

[change_menu_cmd](#)

add_menuitem

Creates a menu item within the specified menu or submenu of the specified window.

Returns nothing.

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 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
```

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 the second argument to the add_menuitem command.

- <text>

(required) Specifies the text to display. Must be the third argument to the add_menuitem command.

- <cmd>

(required) Specifies the command to execute when selecting the menu item with the left mouse button. Must be 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 also echos the results to the Transcript window.

- <shortcut>

(optional) An integer that corresponds to the place of the letter in the menu name to use 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 in the [add_menu](#) example.

Related Topics

[add_menu](#)

[add_menuacb](#)

[add_separator](#)

[add_submenu](#)

[change_menu_cmd](#)

add_separator

Adds a separator as the next item in the specified menu path in the specified window.

Returns nothing.

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 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 the second argument to the add_separator command.

Examples

```
add_separator $wname user
```

This command is also used in the [add_menu](#) example.

Related Topics

[add_menu](#)

[add_menuitem](#)

[add_submenu](#)

[change_menu_cmd](#)

[add_submenu](#)

Creates a cascading submenu within the specified menu path of the specified window. Can be used within a submenu.

Returns the full Tk path to the new submenu widget.

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 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 the second argument to the add_submenu command.
- <name>
(required) Specifies the name to be displayed on the submenu. Must be the third argument to the add_submenu command.
- <shortcut>
(optional) An integer that corresponds to place 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 in the [add_menu](#) example.

Related Topics

[add_menu](#)
[add_menucb](#)
[add_menuitem](#)
[add_separator](#)
[change_menu_cmd](#)

alias

Displays or creates user-defined aliases. Any arguments passed on invocation of the alias are 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 use when invoking the commands.
- "<cmds>"
(optional) Specifies the command or commands to evaluate when invoking the alias.
Specify multiple commands 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 named "myquit" that, when executed, invokes write list to write the contents of the List window to the file *mylist.save*, and then invokes quit to exit Questa SIM.

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

archive load

Enables 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 open 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. If you use a list of file names, you must enclose it in curly braces { }. The name of the wlf file must exactly match the name and file path (if provided) specified in the archive write command.

Related Topics

[archive write](#)

archive write

Enables 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. You can use this archived file to perform post-simulation debugging in a location that differs from where 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 create. 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) Indicates to capture source files in the archive. This is off by default, which means no source is captured in the archive.
- -dbg <dbg_file_name>
(optional) Specifies the name of an existing debug database (.dbg) file to include in the archive.

assertion action

Enables you to set the assertion action for concurrent assertion starts, failures, passes, and antecedent matches.

Syntax

```
assertion action -cond [start | antecedent | pass | fail | failup | faildown | 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> ...]
```

Description

The actions for this command can be: continue, break, exit, or a tcl subroutine call.

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

Restriction

 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 debugging is set – for example, the -assertdebug switch is used with the [vsim](#) command.

Arguments

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

(required) Designates the assertion condition.

 start — When a new assertion attempt starts.

 pass — When an assertion passes.

 fail — When an assertion fails.

 failup —

 faildown —

 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. You can change this limit with the AssertionThreadLimit or CoverThreadLimit variables, set in the *modelsim.ini* file. (Refer to [AssertionThreadLimit](#) or [CoverThreadLimit](#) in the User's Manual.)

global_fail_limit <value> — When the 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 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 take 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 take 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 cannot be combined with this argument. The following options 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 assertions whose names match the specified pattern.

<pattern> — Specifies the character(s) to match 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 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

- In the following command example:

```
assertion action -cond global_fail_limit 100 -exec break
```

a simulation break occurs 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 \[Questa SIM User's Manual\]](#)

assertion active

Instructs the simulator to report on any active assertion directives at the end of simulation (EOS). Active assertion directives are then 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 reports both the usual “active at end of simulation” note and a PSL strong error message.

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 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 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 \[Questa SIM User's Manual\]](#)

assertion count

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.

Syntax

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

Arguments

- **-fails | -failattempts**
(required) Controls reporting of assertion directive failures. Must be 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 | vhndlams | spice**
(optional) Specifies assertions of a specific language. You can specify multiple languages by using this option multiple times. If you do not use this option, all languages are specified by default.
- **-concurrent | -immediate | -soa | -extract | -meas**
(optional) Reports only on the specified type of assertions:
 - concurrent — Concurrent assertions.
 - immediate — Immediate assertions.
 - soa — Save Operating Area assertions in SPICE.
 - extract — Extract Waveform characteristics in SPICE.
 - meas — Measure Waveform Characteristics in SPICE.These are mutually exclusive options. If you do not specify an argument, all 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. Specifying -severity selects only assertions with the same or higher severity level. If not specified, selects assertions of all severities.

- **-recursive**

(optional) *For use with wildcard matching.* Specifies that the scope of the matching is to descend recursively into subregions. Omitting -recursive limits the search to the selected region. Applies to all paths you specify in the command. You can abbreviate this switch to “-r”.

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

(required) A space-separated list of paths specifying the assertions to be affected. Multiple paths and wildcards are allowed. The path specifies assertions or a design region containing multiple assertions. Must be 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 \[Questa SIM User's Manual\]](#)

assertion enable

Enables and disables assertions and cover directives.

Syntax

```
assertion enable -on | -off | -release [-assert | -cover] [-concurrent | -immediate | -soa | -extract | -meas] [-filter <pattern>] [-force] [-lang sva | psl | vhdl | vhdlams | spice]... [-recursive] [-severity info | note | warning | error | failure | fatal] <path> [<path> ...]
```

Arguments

- **-on | -off | -release**

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

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

-off — Disables assertions and cover directives.

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

- **-assert | -cover**

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

- **-concurrent | -immediate | -soa | -extract | -meas**

(optional) Reports only on the specified type of assertions:

-concurrent — Concurrent assertions.

-immediate — Immediate assertions.

-soa — Save Operating Area assertions in SPICE.

-extract — Extract Waveform characteristics in SPICE.

-meas — Measure Waveform Characteristics in SPICE.

These are mutually exclusive options. If you do not specify an argument, all are selected.

- **-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 | psl | vhdl | vhndlams | spice**
(optional) Specifies assertions of a specific language. You can specify multiple languages by using this option multiple times. If you do not use this argument, all languages are specified by default.
- **-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. Specifying -severity selects only assertions with the same or higher severity. If not specified, selects assertions of all severities.
- **<path> [<path> ...]**
(required) A space-separated list of paths, that specifies the assertions to be affected. Must be the final argument to the assertion enable command. Multiple paths and wildcards are allowed. The path specifies assertions or a design region containing multiple assertions.

Description

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

If you do not use the -force argument, assertion enable 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 the one that is issued last.

The -release argument 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).

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

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 \[Questa SIM User's Manual\]](#)

assertion fail

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

Syntax

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

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

exit — Halt simulation and exit Questa SIM.

You can change the default by setting the AssertionFailAction variable in the *modelsim.ini* file. (Refer to [AssertionFailAction](#) in the User's Manual.)

- **-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 **-limit**.

<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 assertions that have not reached their limit. You can change the default by setting the AssertionLimit variable in the *modelsim.ini* file. (Refer to [AssertionLimit](#) in the User's Manual.)

- **-log {on | off}**

(optional) Specifies whether to write a transcript message when an assertion fails. You can specify this argument multiple times; each instance applies to all paths that follow it in the command line.

on — (default) Enable transcript logging for all types of assertion failures (PSL).

off — Disable transcript logging.

You can change the default by setting the AssertionFailLog variable in the *modelsim.ini* file. (Refer to [AssertionFailLog](#) in the User's Manual.)

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 from the GUI or by setting several variables in the *modelsim.ini* file; refer to [IgnoreSVAError](#), [IgnoreSVAFatal](#), [IgnoreSVAInfo](#), and [IgnoreSVAWarning](#) in the Users' Manual.

- **-lvlog**

(optional) Local variable values corresponding to failed assertion threads print to the Transcript when you run **vsim** -assertdebug. You can change the default (on) by setting the AssertionFailLocalVarLog variable in the *modelsim.ini* file. (Refer to [AssertionFailLocalVarLog](#) in the User's Manual.)

- **-print_msg**

(optional) Prints a default message whenever an assertion fails, regardless of what is specified in the action block, and even when no action block is specified. For VHDL assertions, the severity will match that of the specific assertion.

- **-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 specifying the assertions to be affected. Multiple paths and wildcards are allowed. The path specifies assertions, or a design region containing multiple assertions. Must be 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 is 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 \[Questa SIM User's Manual\]](#)

assertion pass

Configures simulator behavior in response to a SystemVerilog or PSL assertion pass.

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. You can specify this option 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 default by setting the AssertionPassLog variable in the *modelsim.ini* file. (Refer to [AssertionPassLog](#) in the User's Manual.)

The -log argument does not apply to SystemVerilog assertion messages coming from action blocks, which are always logged and are controlled by severity system tasks (\$fatal, \$error, \$warning and \$info). You can choose to ignore these system tasks in the GUI or by setting variables in the *modelsim.ini* file; refer to [IgnoreSVAError](#), [IgnoreSVAFatal](#), [IgnoreSVAInfo](#), and [IgnoreSVAWarning](#) in the User's Manual.

- **-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 match in the search. You can use wildcards.

- **-recursive**

(optional) *For use with wildcard matching*. A switch that specifies that the scope of the matching is to descend recursively into subregions. Omitting -recursive limits the search to the selected region. Applies to all paths specified in the command.

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

(required) Specifies the assertions to be affected. Specify multiple paths as a space-separated list. Must be the final argument to the assertion pass command. You can use multiple paths and wildcards. The path specifies assertions or a design region containing multiple assertions.

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 \[Questa SIM User's Manual\]](#)

assertion profile

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.

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 any time the number of threads in an assertion or cover directive exceeds the thread threshold at a clock edge. Functions only 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. Must be the final argument to the assertion profile command. You can give assertion profile on at any time during simulation.
 on — Collection of memory profile data enabled.
 off — Collection of memory profile data disabled.

Description

The assertion profile command reports fine-grained memory usage profile data in the Assertions Browser and Cover Directives Browser, in four columns: current memory, peak memory, peak memory time, and cumulative threads.

If you use assertion profile in the middle of the simulation, memory usage profiling starts/stops from that simulation time onwards. The command does not require that you compile designs with +acc, or that you run vsim in the -assertdebug mode. You can give the command in a fully optimized simulation run.

Examples

- Thread threshold is reached.

```
assertion profile -threadthreshold 100 on
```

Reaching the threshold produces the following transcript message:

```
# ** 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 match 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 \[Questa SIM User's Manual\]](#)

[Assertions Window \[Questa SIM GUI Reference Manual\]](#)

[Cover Directives Window \[Questa SIM GUI Reference Manual\]](#)

assertion report

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 | vhdlams | spice] [-noconcurr] [-noimmed] [-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. You can use environment variables in the pathname.
- **-filter <pattern>**
 <pattern> — Specifies the character(s) to match in the search. Permits wildcards.
- **-lang psl | sva | vhdl | vhdlams | spice**
(optional) Specifies assertions of a specific language. You can specify multiple languages by using this option multiple times. If you do not use this option, all languages are specified by default.
- **-noconcurr**
(optional) Disables reporting of any concurrent assertions.
- **-noimmed**
(optional) Disables reporting of any immediate assertions.
- **-nosoa**
(optional) Disables reporting of any SPICE Safe Operating Area assertions.
- **-noextract**
(optional) Disables reporting of any SPICE Extract Waveform Characteristics.
- **-nameas**
(optional) Disables reporting of any SPICE Measure Waveform Characteristics.

- **-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. You can use 'assertion profile on/off' to switch the profiling for assertions on or 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. Permits wildcards.

Examples

Consider the following example test module, which contains four assertions: b1, b2, c1, and c2.

```
module test;
reg a, b;

reg clk;
initial clk =0;
always #50 clk = ~clk;

b1:assert property (@(posedge clk) a|=>b);
b2:assert property (@(posedge clk) a|=>b);
c1:assert property (@(posedge clk) a|=>b);
c2:assert property (@(posedge clk) a|=>b);

initial
begin
  a=0; b=1;
#200 a=1; b=1;
#100 a=0; b=1;
#500 $finish;
end

endmodule
```

You can use the assertion report command to report on all assertions.

```
assertion report *
```

This produces the following report:

```
#-----
#Name      File (Line)          Failure Count   Pass Count
#-----
#/test/b1  src/miscreport01.sv(9)    0            0
#/test/b2  src/miscreport01.sv(10)   0            0
#/test/c1  src/miscreport01.sv(11)   0            0
#/test/c2  src/miscreport01.sv(12)   0            0
#
```

If you are interested in seeing a report on only assertion b1, use the **-filter b1** argument with the assertion report command.

assertion report * -filter b1

This produces the following report:

```
#-----
#Name      File (Line)          Failure Count   Pass Count
#-----
#/test/b1  src/miscreport01.sv(9)   0            0
#
```

You can also use wildcards with the **-filter** argument.

assertion report * -filter c*

This produces the following report:

```
#-----
#Name      File (Line)          Failure Count   Pass Count
#-----
#/test/c1  src/miscreport01.sv(11)  0            0
#/test/c2  src/miscreport01.sv(12)  0            0
#
```

Related Topics

[assertion action](#)
[assertion active](#)
[assertion count](#)
[assertion enable](#)
[assertion fail](#)
[assertion profile](#)
[atv log](#)

[Configuring Assertions \[Questa SIM User's Manual\]](#)

[Assertions Window \[Questa SIM GUI Reference Manual\]](#)

[Cover Directives Window \[Questa SIM GUI Reference Manual\]](#)

atv log

Prerequisites:

- Run the [vopt](#) command with the `+acc=a` argument or the [vsim](#) command with `-voptargs+=acc=a` argument.
- Run the [vsim](#) command with the `-assertdebug` argument.

Enables or disables assertion thread viewing (ATV) for the specified assertion or assertions. Specify multiple assertions by their pathnames.

Syntax

```
atv log {-enable | -disable} [-asserts] [-covers] [-recursive] <path> [<path> ...]
```

Arguments

- [-enable](#) | [-disable](#)
(required) Controls assertion thread viewing for the specified assertions. Must be 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.* 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 affected assertions. Must be the final argument to the [atv log](#) command. You can use wildcards. Each path specifies an assertion or a design region containing multiple assertions.

Related Topics

- [assertion action](#)
- [assertion active](#)
- [assertion count](#)

[assertion enable](#)

[assertion fail](#)

[assertion pass](#)

[assertion profile](#)

[Configuring Assertions \[Questa SIM User's Manual\]](#)

[Viewing Assertions in the Assertions Window \[Questa SIM User's Manual\]](#)

batch_mode

Returns “1” if Questa SIM is operating in batch mode, otherwise it returns “0.” 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 works both in and out of batch mode, use the batch_mode command to determine which command to use. For example:

```
if [batch_mode] {  
    log /*  
} else {  
    add wave /*  
}
```

Related Topics

[General Modes of Operation \[Questa SIM User's Manual\]](#)

bd

Deletes a breakpoint. You can delete multiple breakpoints by specifying separate information groupings on the same command line.

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 to delete the breakpoint. The filename must match the one used to set the breakpoint, including whether you used a full pathname or a relative name. Must be the first argument to the bd command, if you do not specify <id_number> or <label>.
- <line_number>
(required) A string that specifies the line number of the breakpoint to delete.
- <id_number> | <label>
(required when not specifying <filename>.) Specifies the identification of breakpoints using markers assigned by the [bp](#) command. Must be the first argument to the bd command, if you do not specify <filename>.

 <id_number> — A string that specifies the identification number of the breakpoint to delete. Use the -id argument to the [bp](#) command to set the identification number. 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 delete. Use the -label switch to the [bp](#) command to set the label.

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](#)

bookmark add wave

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.

Note

 You can also interactively add a bookmark through the GUI by selecting the **Wave > Bookmarks > Bookmarks** menu item.

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 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 ({}).

You must also enclose the complete grouping of <range_start> and <range_end> 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 specifying the vertical scroll position of the window. You must specify a zoom range before you specify topindex. The number identifies which object to scroll the window to. For example, specifying 20 will scroll the Wave window down to show the 20th object.

-
- **-window <window_name>**

(optional) Specifies the window to add the bookmark to. If you omit this argument, the bookmark is added to 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](#)

bookmark delete wave

This command deletes bookmarks from the specified Wave window.

Note

 You can also interactively delete a bookmark through the GUI by selecting the **Wave > Bookmarks > Bookmarks** menu item.

Syntax

```
bookmark delete wave {<label> | -all} [-window <window_name>]
```

Arguments

- <label> | -all

(required) Controls which bookmarks to delete. Must be the first argument to the bookmark delete wave command.

<label> — Specifies the name of the bookmark to delete.

-all — Specifies to delete all bookmarks in the window.

- -window <window_name>

(optional) Specifies the window in which to delete the bookmark. If you omit this argument, the command deletes bookmark(s) in the current default Wave window.

<window_name> — The name of the wave window containing the bookmark to delete.

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](#)

bookmark goto wave

Zooms and scrolls a Wave window using the specified bookmark.

Note

 You can also interactively navigate between bookmarks through the GUI by selecting the **Wave > Bookmarks > Bookmarks** menu item.

Syntax

`bookmark goto wave <label> [-window <window_name>]`

Arguments

- `<label>`
(required) Specifies the bookmark to go to. Must be the first argument to the bookmark goto wave command.
- `-window <window_name>`
(optional) Specifies the Wave window to which the bookmark applies. You can use a bookmark only in the window it was created in.

`<window_name>` — The name of the wave window containing the specified bookmark.

Related Topics

[bookmark add wave](#)

[bookmark delete wave](#)

[bookmark list wave](#)

bookmark list wave

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. You can use bookmarks only in the window in which they were originally created.

Related Topics

[bookmark add wave](#)

[bookmark delete wave](#)

[bookmark goto wave](#)

bp

Either sets a file-line breakpoint, or returns information about breakpoint(s). Allows condition expressions to use enum names, as well as literal values.

Note

 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

 Your 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 “[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} [-ancestor] [-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> [{<start_line_number> <end_line_number>} | <line_number>]]
```

Reporting all breakpoints

```
bp
```

Note

 Specifying `bp` with no arguments returns a list of all breakpoints in the design, containing information about each breakpoint. For example, the command “`bp`” might return:

```
# 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 uses the source file of the current context.

- <line_number>
(required to set an HDL breakpoint) Specifies the line number on which to set the breakpoint.
- in
(required for task or function breakpoints) Supports the lookup of Verilog and SystemVerilog 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. Does not work for VHDL or SystemC.
- -c
(required to set a C breakpoint) Applies the bp command and its arguments to SystemC instances in the design. Must be 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 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.
- -ancestor
(optional) Stops the simulation only when an ancestor parent of the process matches the given process-name.
- -appendinst
(optional) When specifying multiple breakpoints with -inst, appends 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. The bp command enables breakpoints by default.
- <command>...
(optional, must be specified as the final argument) Specifies one or more commands to execute 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 also applies if you use a DO file script within the command string.

You cannot use a [restore](#) command.

If you need to enter many commands 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 behavior differs from 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 a condition 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:

Operator on	Operator Syntax
equals	<code>==, =</code>
not	<code>!=, /=</code>
equal	
AND	<code>&&, AND</code>
OR	<code> , OR</code>

The operands can be object names, signal name's event, or constants. Subexpressions in parentheses are permitted. The command is 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> [{<start_line_number>} <end_line_number>] | <line_number>]`
(optional) Returns information about the breakpoint(s) and the list of executable lines in the specified file. When specified without an argument, the command returns the list of all breakpoints and its status. Returns nothing if the line number(s) specified in the argument are not executable.

`<filename>` — The name of the file you want to query for breakpoint(s).

`<start_line_number> <end_line_number>` — Specifies a range of line numbers. The command returns a list of any executable lines within the range.

`<line_number>` — The line number you want to query for a breakpoint.

When `<line_number>` is an argument, then the command returns six fields of information.

For example, the command:

```
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 exist.
 - 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 numbers, 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 all executable lines in testadd.vhd between lines 2 and 59.

bp -query testadd.vhd 2 59

- List details about the breakpoint on line 48.

bp -query testadd.vhd 48

- 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"

Related Topics

[add button](#)

[bd](#)

[Editing File-Line Breakpoints \[Questa SIM GUI Reference Manual\]](#)

call

Calls SystemVerilog static functions and class functions (but not tasks) from the vsim command line in live simulation mode. Calls PLI and VPI system tasks and system functions. Returns function return values to the vsim shell as a Tcl string. Returns class references as class instance IDs.

Syntax

Note

 The grouping of arguments can vary depending on how you want the command to perform a search on a system task or function name. For example:

- <pathToFunction> <classInstancePath> —SystemVerilog static functions and class functions
- -usertf -builtin <systfName> — PLI and VPI system tasks and system functions (restricts the search for the task/function name to either the user-added PLI/VPI routines or to the built-in routines):

```
call [-env <hierEnvPath>] [{<pathToFunction> [<classInstancePath>]}] [{-usertf | -builtin <systfName>}] [<arg1> [<arg2>] ...[<argN>]]
```

Arguments

- -env <hierEnvPath>
(optional) Hierarchical environment path to use as the starting scope for the object name lookups. If present, must appear before actual function name.
- <pathToFunction>
(required when calling a System Verilog static or class function) The name of a function, which you can specify by:
 - The path to the function declaration, through the structural hierarchy, or declaration hierarchy. You must specify hierarchical paths must 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).
 - A class instance hierarchical path.
 - A class instance id string.
- <classInstancePath>
(optional) Must be specified if the function path is a declaration path and the function is a non-static class function. Do not specify the class instance path name if the given function path is a class instance variable reference or a class instance name in the format @<class_type>@nnn, because the class instance information can be extracted from the pathname itself.

- **-builtin**
(optional) Search only built-in system functions or task names. (The \$ is understood to be a prefix.)
- **-userf**
(optional) Search only user-defined system functions or task names. (The \$ is understood to be a prefix.)
- **<systfName>**
(required when calling a system task or function) The system task or function to execute. You can specify the sysfName according to the following syntax rules:
 - `\$<systfName>` (for example, `\$display` or `\$mytask`)
 - With `-userf` or `-builtin` flags (as the case may be). (for example, `-userf mytask` or `-builtin display`)
 - **<systfName>** (for example, `display` or `mytask`) In this case SystemVerilog static functions and class functions are searched first for a match, and then the PLI/VPI system tasks/functions.
- **<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 can 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. Strings containing spaces or special characters must be enclosed in quotation marks (" ") 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.

Examples

Calling a System Verilog Static or 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) => ab)

- 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
```

capstats

Generates a detailed report of memory usage in your design.

Note

 The capstats report, with any allowed arguments, is more accurate when used in conjunction with the **vsim** -capacity argument.

Usage

```
capstats [-du [+summary | +details | +duname= ]] [-decl] [-line] [<filename>]
```

Arguments

- **none**
Generates a summary report under the top-level categories.
- **-du**
When used in conjunction with the vsim -capacity argument, generates a design unit based report of memory use.
 - +summary — provides a summary report for the memory usage within a design unit.
 - +details — provides a more expansive report that includes details about dynamic allocations.
 - +duname=<duname> — enables you to generate a report that covers only the specified design unit. You can use +duname=<duname> multiple times to specify multiple design units.
- **-decl**
When used in conjunction with the vsim -capacity argument, generates a declaration based report for SV dynamic objects.
- **-line**
When used in conjunction with the vsim -capacity=line argument, generates a line based report for SV dynamic objects.
- **<filename>**
Prints the report to the specified file. If you do not specify a filename, the report prints in the transcript file.

cd

Changes the Questa SIM local directory to the specified directory.

Syntax

cd [[<dir>](#)]

Arguments

- <dir>

(optional) Specifies a full or relative directory path for Questa SIM to use as the local directory. If you do not specify a directory, the command changes to your home directory.

Description

Executing a **cd** command closes the current project. You cannot execute this command while a simulation is in progress.

cdbg

This command provides command-line equivalents of the menu options that are available for C Debug.

Note

 Some cdbg commands require the "on | off" argument. 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 called user function. 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 finishes loading.

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 your application calls a named FLI or PLI function. 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 \[Questa SIM User's Manual\]](#)

[Debugging Functions when Quitting Simulation \[Questa SIM User's Manual\]](#)

[Finding Function Entry Points with Auto Find bp \[Questa SIM User's Manual\]](#)

[Identifying All Registered Function Calls \[Questa SIM User's Manual\]](#)

change

This command modifies the value of a: VHDL constant, generic, or variable; Verilog register or variable; or C variable if running C Debug.

Syntax

`change <variable> <value>`

Description

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 all uses of deferred (or other) constants, because the simulation depends on constants not changing after elaboration. You must treat VHDL constants as immutable after compilation.

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.

Supported objects include:

- VHDL
 - Scalar variable, constant, and 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 sub-element 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.
- Sub-elements 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.
 - Not supported — SystemC primitive channels.
- <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 does 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 "my string," 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 in curly braces because of the space between the value and the units.

change mytimegeneric {"500 ps"}

change_menu_cmd

Changes the command executed by a specified menu item label, in a specified menu, in a specified window. Returns nothing.

Note

 This command functions only on already existing menu paths and labels.

Syntax

`change_menu_cmd <window_name> <menu_path> <label> <cmd>`

Arguments

The arguments `<window_name>`, `<menu_path>`, `<cmd>`, and `<label>` are order dependent. You must enter them in the order shown in the Syntax section.

- `<window_name>`
(required) Specifies the Tk path of the window containing the menu. You must use double quotes ("") in place of `<window_name>` to identify the Main window. 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 execute when the menu item is selected.

Related Topics

[add_menu](#)
[add_menucb](#)
[add_menuitem](#)
[add_separator](#)
[add_submenu](#)

check contention add

Enables contention checking on the specified nodes. Allowed nodes are Verilog nets and VHDL signals of types std_logic and std_logic_vector. Ignores any other node types, and nodes that do not have multiple drivers.

Syntax

```
check contention add {[ -in ] [ -out ] [ -inout ] | [ -ports ]} [ -internal ] [ -r ] <node_name>...
```

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
0	0	yes
1	Z	no
0	1	yes
L	1	no
L	H	yes

Detection of a bus fight returns an error message that specifies the node and its drivers' current driving values. If a node's drivers later change value, and the node is still in contention, a new message gives the new values of the drivers. A message is also issued when the contention ends. You can use the bus contention checking commands on VHDL and Verilog designs.

You can set a time limit (the default is zero) for contention checking with the -time <limit> argument to the check contention config command. If you choose to modify the limit, do so prior to invoking any check contention add commands.

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 to enable contention checking recursively into subregions. If omitted, enables contention checking to only the current region.
- **<node_name>...**
(required) Specifies the name of a node. Must be the final argument to the check contention add command.

Related Topics

[check contention config](#)

[check contention off](#)

check contention config

Allows you to write checking messages to a file. By default, any messages display on your screen. Also allows you to configure the contention time limit.

Syntax

```
check contention config [-file <filename>] [-time <limit>[<unit>]]
```

Arguments

- **-file <filename>**

(optional) Writes contention messages to the specified file instead of displaying them on the screen.

<filename> — The name of a file where contention messages are saved.

- **-time <limit>[<unit>]**

(optional) Specifies a limit to the time that a node can be in contention. If a node's contention time meets or exceeds the limit, contention is detected.

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

<unit> — (optional) A suffix specifying a unit of time. If you specify <unit>, you must enclose <limit> and <unit> within curly braces ({ }). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. The default is to specify the current simulation resolution by omitting <unit>.

Related Topics

[check contention add](#)

[check contention off](#)

check contention off

Disables contention checking for the specified nodes.

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) Disables contention checking recursively into subregions. If omitted, contention check disabling is limited to the current region.
- **<node_name> ...**
(required) Specifies the named node(s). Must be the final argument to the check contention off command.

Related Topics

- [check contention add](#)
- [check contention config](#)

check float add

Enables float checking for the specified nodes.

Syntax

```
check float add {[ -in ] [ -out ] [ -inout ] | [ -ports ]} [ -internal ] [ -r ] <node_name>...
```

Description

Bus float checking detects nodes that are in a high impedance state for a time period that meets or exceeds a user-defined limit. Detection of a float violation results in an error message identifying the node. A message is also issued when the float violation ends. You can use the bus float checking commands on VHDL and Verilog designs.

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 with the -time <limit> argument to the check float config command. If you choose to modify the limit, do so prior to invoking any check float add commands.

Arguments

- **-r**
(optional) Enables float checking recursively into subregions. If omitted, limits float checking 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 the final argument to the check float add command.

Related Topics

[check contention config](#)

check float off

check float config

Enables you to write checking messages to a file (messages display on your screen by default). Also enables you to 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 you select this option, messages do not display on the screen.

`<filename>` — Specifies the name of the file to save float messages to.

- `-time <time><unit>`

(optional) Specifies the length of time that a node can be floating. Detects an error if a node meets or exceeds the float time limit. Configure this 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 the time unit. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. The default is to specify the current simulation resolution by omitting `<unit>`. If you specify `<unit>`, you must enclose `<time>` and `<unit>` within curly braces ({}).

Related Topics

[check float add](#)

[check float off](#)

check float off

Disables float checking for the specified nodes.

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) Disables float checking recursively into subregions. If omitted, disables float checking in only the current region.
- **<node_name> ...**
(required) Disables checking for the named node(s). Must be the final argument to the check float off command.

Related Topics

[check float add](#)

[check float config](#)

check stable off

Disables stability checking. Clock cycle numbers and boundaries are still tracked, and you can later re-enable stability checking with check stable on.

Syntax

`check stable off`

Arguments

None

Related Topics

[check stable on](#)

check stable on

Enables stability checking on the entire design.

Syntax

```
check stable on [-period <time> [<unit>]] [-file <filename>] [-strobe <time> [<unit>]]
```

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 that period during which the circuit must be stable. If there are pending driver events at the strobe time, an error message informs you that a violation has been detected. The message identifies the driver that has a pending event, the node that it drives, and the cycle number. You can use design stability checking commands on VHDL and Verilog designs.

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 the first argument to the check stable on command.

<time> — Any non-negative integer.

<unit> — (optional) A suffix specifying a unit of time. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit>, you must enclose <time> and <unit> within curly braces ({}).

This argument is not required if you later re-enable checking after it was disabled with the [check stable off](#) command.

- **-file <filename>**

(optional) Writes the error messages to a specified file. If you select this option, messages do not display to the screen.

<filename> — Specifies the name of the file to save error messages to.

- **-strobe <time> [<unit>]**

(optional) Specifies the time to wait during each clock cycle before performing the stability check. The default strobe time is the time specified in -period. If the strobe time falls on a period boundary, the check is actually performed one timestep earlier. Normal practice is to specify the strobe time 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. The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit>, you must

enclose <time> and <unit> within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

Examples

- Performs a stability check 99 ps into each even-numbered clock cycle (where cycle numbers start at 1).

check stable on -period "100 ps" -strobe "199 ps"

Related Topics

[check stable off](#)

checkpoint

Saves the state of your simulation as a checkpoint file that you can restore.

Syntax

```
checkpoint [-dir] <pathname>  
checkpoint <pathname>/[<filename>]
```

Description

The states of simulation that you can save include:

- *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).

During a simulation, you can use a saved checkpoint file with the [restore](#) command to restore the simulation to a previous state. You can also use the vsim -restore command to start a vsim session with a checkpoint file.

Compression of a checkpoint file is controlled by the CheckpointCompressMode variable in the *modelsim.ini* file. (Refer to [CheckpointCompressMode](#) in the User's Manual.)

If a checkpoint occurs while Questa SIM is writing a VCD file, the entire VCD file gets copied into the checkpoint file. Because VCD files can be very large, disk space problems could occur. Consequently, Questa SIM issues a warning in this situation.

If you checkpoint 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 run 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.
- For most designs containing SystemC, the checkpoint command is not supported once the design has been loaded. However, some SystemC modules do support the use of the checkpoint command with `vsim -scchkpntrestore`. For more information, refer to [Checkpoint and Restore in a SystemC Design](#) in the User's Manual.

Arguments

- **-dir**
(optional) Creates a directory with the specified pathname for containing checkpoint data. The two valid scenarios for specifying a directory with the <pathname> are:
 - The "-dir" option is specified with a <pathname>. The command creates a new directory if the specified directory does not exist.
 - The "-dir" option is not specified with the <pathname>. The <pathname> represents a pre-existing directory.In both of these cases, the command creates a a checkpoint file named '*vsim.cpt*' in the specified directory. In addition, in both of these cases, the command stores other checkpoint data in the specified directory. The checkpoint data can be in more than one file.
- **<filename>**
(optional) Specifies the name of the checkpoint file. When <filename> is not specified, the default *vsim.cpt* filename is used.
- **<pathname>**
(required) Specifies the pathname of the directory where the command saves the *vsim.cpt* checkpoint file. In addition, specifies the pathname of the checkpoint directory when “-dir” is present, or the pathname of the single checkpoint filename when “-dir” is absent.

Examples

checkpoint -dir <pathname>

This syntax creates the directory named <pathname> in the current directory. <pathname> can be hierarchical. You cannot specify the name of the CPT file; it is always *vsim.cpt*. For example:

- `checkpoint -dir foo` — creates the directory "foo" in the current directory as well as the *vsim.cpt* checkpoint file.
- `checkpoint -dir 1/2/3` — creates the hierarchy "1/2/3/" in the current directory as well as the *vsim.cpt* checkpoint file.

- `checkpoint -dir ..`/test — creates the directory one level above the current directory as well as the `vsim.cpt` checkpoint file.

checkpoint <pathname>/[<filename>]

This syntax assumes the directory structure specified by <pathname> exists and creates the CPT file named <filename> in that directory. If you do not specify <filename> the command creates a checkpoint file with the default filename, `vsim.cpt`, in the <pathname> directory.

- `checkpoint foo` — creates a checkpoint file with the default filename `vsim.cpt` in the foo/ directory.
- `checkpoint foo/bar` — creates a checkpoint file with the default filename `vsim.cpt` in the foo/bar/ directory.
- `checkpoint foo/test.cpt` — creates a checkpoint file named `test.cpt` in the foo/ directory.
- `checkpoint test.cpt` — creates a checkpoint file named `test.cpt` in the current directory.

Related Topics

[Checkpointing and Restoring Simulations \[Questa SIM User's Manual\]](#)

classinfo ancestry

Returns class inheritance hierarchy for a named class type.

Syntax

```
classinfo ancestry [-dataset <name>] [-n] [-o <outfile>] [-tcl] <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 to write output to.
- -tcl
(optional) Returns a tcl list instead of formatted output.

Related Topics

[ClassDebug \[Questa SIM User's Manual\]](#)

[classinfo descriptive](#)

[classinfo find](#)

[classinfo instances](#)

[classinfo isa](#)

[classinfo report](#)

[classinfo stats](#)

[classinfo trace](#)

[classinfo types](#)

classinfo descriptive

Returns the descriptive class name for the specified authoritative class name.

Syntax

```
classinfo descriptive [-dataset <name>] [-exact | -glob | -regexp] [-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.
- -o <outfile>
(optional) Sends the results of the command to <outfile> instead of the transcript.
 <outfile> — Specifies the name of the file to write the output to.
- -regexp
(optional) Treats <class_name> as a regular expression.
- -tcl
(optional) Returns a tcl list instead of formatted output.

Related Topics

[ClassDebug \[Questa SIM User's Manual\]](#)

[Logging Class Types and Class Instances \[Questa SIM User's Manual\]](#)

[Working with Class Types \[Questa SIM User's Manual\]](#)

[Analyzing Class Types \[Questa SIM User's Manual\]](#)

[classinfo ancestry](#)

[classinfo find](#)

[classinfo instances](#)

[classinfo isa](#)

[classinfo report](#)

[classinfo stats](#)

[classinfo trace](#)

[classinfo types](#)

classinfo find

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 to write the output to.
- -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 \[Questa SIM User's Manual\]](#)

[classinfo ancestry](#)

[classinfo descriptive](#)

[classinfo instances](#)

classinfo find

[classinfo isa](#)

[classinfo report](#)

[classinfo stats](#)

[classinfo trace](#)

[classinfo types](#)

classinfo implements

Displays a list of the classes which implement a specified SystemVerilog interface class. The type of the class argument affects the contents of the list.

Syntax

```
classinfo implements [-dataset <name>] [-tcl] [-o <outfile>] <class_type>
```

Arguments

- <class_type>

(required) Name of the SystemVerilog class 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.

- -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 \[Questa SIM User's Manual\]](#)

[classinfo descriptive](#)

[classinfo find](#)

[classinfo instances](#)

[classinfo interfaces](#)

[classinfo isa](#)

[classinfo report](#)

[classinfo stats](#)

[classinfo trace](#)

[classinfo types](#)

classinfo instances

Reports the list of existing class instances of a specific class type. You can use this to determine the class instances to log or examine, and to debug excessive memory use problems caused by class instances that are not cleaned up properly.

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 to write the output to.

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 \[Questa SIM User's Manual\]](#)

[classinfo ancestry](#)

[classinfo descriptive](#)

[classinfo find](#)

[classinfo implements](#)

[classinfo interfaces](#)

[classinfo isa](#)

[classinfo report](#)

[classinfo stats](#)

[classinfo trace](#)

[classinfo types](#)

classinfo interfaces

Lists the interface class types that 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 to write the output to.
- -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
```

- Used with no argument, the command returns the names of all interface classes:

```
vsim> classinfo interfaces
```

Output list:

```
# /test8/M
# /test8/N
# /test8/X
# /test8/Y
```

Related Topics

[ClassDebug \[Questa SIM User's Manual\]](#)

[classinfo descriptive](#)

[classinfo find](#)

[classinfo implements](#)

[classinfo instances](#)

[classinfo isa](#)

[classinfo report](#)

[classinfo stats](#)

[classinfo trace](#)

[classinfo types](#)

classinfo isa

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 to write the output to.
- -tcl
(optional) Returns a tcl list instead of formatted output.

Related Topics

[ClassDebug \[Questa SIM User's Manual\]](#)

[classinfo ancestry](#)

[classinfo descriptive](#)

[classinfo find](#)

[classinfo instances](#)

[classinfo report](#)

[classinfo stats](#)

[classinfo trace](#)

[classinfo types](#)

classinfo report

Prints detailed reports on class instance usage. Displays columns for class type names and their current, peak and total class instance counts.

Syntax

```
classinfo report [-c [fntp]] [-dataset <name>] [-m <maxout>] [-o <outfile>]  
[-sort [a | d] [f | n | t | p | c]] [-tcl] [-z]
```

Arguments

- **-c [fntp]**
(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 path name.
 - n — The Class Name column displays the class name, and for parameterized classes, it displays the internally generated class specialization name.
 - 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 simultaneously 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 to the transcript.
 - <outfile> — Specifies the name of the file to write the output to.
- **-sort [a | d] [f | n | t | p | c]**
(optional) Specifies whether to sort report information in ascending or descending order, and which column to sort by. You can specify only one column 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 that have 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 the 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 \[Questa SIM User's Manual\]](#)
- [classinfo ancestry](#)
- [classinfo descriptive](#)
- [classinfo find](#)
- [classinfo instances](#)
- [classinfo isa](#)
- [classinfo stats](#)
- [classinfo trace](#)
- [classinfo types](#)

classinfo stats

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 to write the output to.
- `-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 \[Questa SIM User's Manual\]](#)

[classinfo ancestry](#)

[classinfo descriptive](#)

[classinfo find](#)

[classinfo instances](#)

[classinfo isa](#)

[classinfo report](#)

[classinfo trace](#)

[classinfo types](#)

classinfo trace

Displays the active references to the specified class instance. 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 can lead to uncovering bugs in class reference management, and 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 to write the output to.
- -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 \[Questa SIM User's Manual\]](#)

[classinfo ancestry](#)

[classinfo descriptive](#)

[classinfo find](#)

[classinfo instances](#)

[classinfo isa](#)

[classinfo report](#)

[classinfo stats](#)

[classinfo types](#)

classinfo types

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 to write the output to.
- -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 \[Questa SIM User's Manual\]](#)

[classinfo ancestry](#)

[classinfo descriptive](#)

[classinfo find](#)

[classinfo implements](#)

[classinfo instances](#)

[classinfo interfaces](#)

[classinfo isa](#)

[classinfo report](#)

[classinfo stats](#)

[classinfo trace](#)

compare add

Creates an object that is a comparison of signals in a reference design to 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. Ignores constant signals such as parameters and generics.

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>
```

Description

Refer to “[Waveform Compare](#)” in the User’s Manual 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 the results of comparisons 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, see the -vhdlmatches argument). Since you can also

set state matching 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 the local matching values specified when the comparison was created with compare add, or subsequently configured with [compare configure](#).
1. If no local values were set, use global matching values set with the compare options command.
1. If no compare options were set, use default matching values specified by PrefCompare Tcl variables.

Arguments

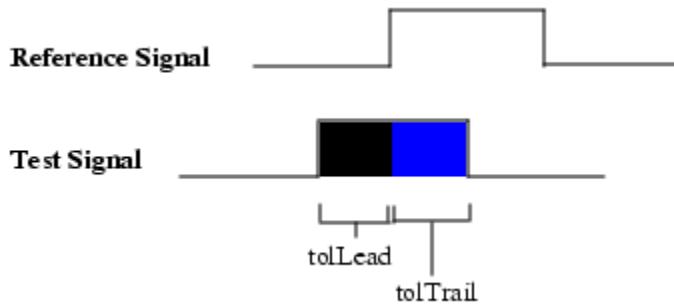
- **-all**
(optional) Specifies comparison of all signals.
- **-help**
(optional) Lists the description and syntax for the compare add command in the Transcript window.
- **-in**
(optional) Specifies comparison of ports of mode IN. You can specify multiple modes in the same command.
- **-inout**
(optional) Specifies comparison of ports of mode INOUT. You can specify multiple modes in the same command.
- **-out**
(optional) Specifies comparison of ports of mode OUT. You can specify multiple modes 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. You can specify multiple modes in the same command.
- **-label <label>**
(optional) Specifies the name to display when the comparison is shown in the Wave window.
 <label> — The name for the comparison.
- **-list**
(optional) Displays specified comparisons 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. Ignored if a signal is found that has the same name as the reference signal. When rebuilding the test signal, the name of the reference signal is used as the wildcard prefix.
- **-recursive**
(optional) Specifies to also select signals in all nested subregions, recursively.
- **-separator <string>**
(optional) Used with the **-rebuild** argument. Specifies a separator to insert 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 use as a separator between the base bus name and the bit indication. The default is an underline “_”.
- **<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 to allow a test signal edge 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. 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 you specify **<unit>**, you must enclose **<time>** and **<unit>** within curly braces ({}).
- **-tolLead <time> [<unit>]**
(optional) Specifies the maximum amount of time to allow a test signal edge 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. 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 you specify **<unit>**, you must enclose **<time>** and **<unit>** within curly braces ({}).
- **-tolTrail <time> [<unit>]**
(optional) Specifies the maximum amount of time to allow a test signal edge to trail a reference edge in an asynchronous comparison.

<time> — Any non-negative integer. The default is 0.

<unit> — (optional) A suffix specifying a unit of time. 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 you specify <unit>, you must enclose <time> and <unit> within curly braces ({}).

Figure 2-1. Graphical representation of tolLead and tolTrail - compare add



- **-verbose**
(optional) Prints the signals selected for comparison, and any type conversions employed, in the Transcript window.
- **-vhdlmatches {<ref-logic-value>=<test-logic-value>:....}**
(optional) Specifies how to match VHDL signal states in the reference dataset to values in the test dataset. Specify values in a colon-separated list of match values. For example:

```
-vhdlmatches {X=XUD:Z=ZD:1=1HD}
```

Default is:

```
{U=UWDXD:X=UWDXD:0=0LD:1=1HD:Z=ZD:W=UWDXD: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. Specify values 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 to add compare signals automatically to the default Wave window.
(default)
- **-wavepane <n>**
(optional) Specifies the pane of the Wave window in which to place the differences.

<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 to add objects.

- -nowin

(optional) Specifies not to add compare signals to any window. 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 the path to the reference signal or region. Can be an absolute path, a relative path, or a glob expression. Relative paths are relative to the current context of the reference dataset. A glob expression matches signals only in the containing context. Must be the final argument to the compare add command.

Examples

- Select signals in the reference and test dataset top region according to the default mode. This example uses asynchronous comparison with the default tolerances, and 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 to sample the signal *gold*::*top.s1* at *clock12*, and compare it with *test*::*top.s1*, also sampled at *clock12*.

compare add -clock clock12 gold::top.s1

- Specify to asynchronously compare signal *gold*::*asynch/abc/s1* to 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 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](#)

compare annotate

Either flags a comparison difference as ignore, or adds a text string annotation to the difference. The text string appears when viewing the comparison difference in info popups or in the output of a compare info command.

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 appears wherever you view the difference.
- **<idNum1> <idNum2>...**
(required) Identifies the difference number to annotate. You can obtain a difference number from the [compare start](#) command, or from the popup window that appears when you place the cursor over the logged difference in the Wave window. Difference numbers are ordered by time of the difference start, but more than one difference can start at a given time. Specify multiple id numbers as a space-separated list. Must be 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 \[Questa SIM User's Manual\]](#)

compare clock

Defines a clock to use for clocked-mode comparisons. In clocked-mode comparisons, signals are sampled and compared at or just after a specified signal edge.

Syntax

```
compare clock [-delete] [-offset <delay>[<unit>]] [-rising | -falling | -both]
               [-when {"<expression>"})} {<clock_name>} <signalization>
```

Arguments

- **-delete**
(optional) Deletes an existing compare clock.
- **-offset <delay>[<unit>]**
(optional) Specifies how long to delay 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. If <unit> is specified, you must enclose <delay> and <unit> within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr. The default is to specify the current simulation resolution by omitting <unit>.
- **-rising**
(optional) Specifies to use the rising edge of the specified signal. (default)
- **-falling**
(optional) Specifies to use the falling edge of the specified signal. The default is rising.
- **-both**
(optional) Specifies to use both the rising and the falling edge of the specified signal. 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 "["GUl_expression_format"](#)" for legal expression syntax.
 "<expression>" — The conditional expression to evaluate. You must enclose the condition expression within quotation marks ("").
- **<clock_name>**
(required) A name for this clock definition. This name is used with the [compare add](#) command when doing a clocked-mode comparison. Must precede the <signal_path> argument.

- <signalization>

(required) A full path to the signal whose edges are to be used as the strobe trigger. Must be 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 \[Questa SIM User's Manual\]](#)

compare configure

Modifies options for compare signals and regions. Applies the modified options to all objects in the specified compare path.

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. You cannot use this switch with the -tol, -tolLead, and -tolTrail options.

<name>
- **-recursive**
(optional) Specifies to also select signals in all nested subregions, and subregions of those, and so forth.
- **-tol <time>[<unit>]**
(optional) Specifies the default maximum amount of time to allow the test signal edge 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. The default is to specify the current simulation resolution by omitting *<unit>*. If you specify *<unit>*, you must enclose *<time>* and *<unit>* within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.
- **-tolLead <time>[unit]**
(optional) Specifies the maximum amount of time to allow a test signal edge 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. The default is to specify the current simulation resolution by omitting *<unit>*. If you specify *<unit>*, you must enclose *<time>* and *<unit>* within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.
- **-tolTrail <time>[unit]**
(optional) Specifies the maximum time to allow a test signal edge to trail a reference edge in an asynchronous comparison.

<time> — Any non-negative integer. The default is 0.

<unit> — (optional) A suffix specifying a unit of time. The default is to specify the current simulation resolution by omitting **<unit>**. If you specify **<unit>**, you must enclose **<time>** and **<unit>** within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

- **-vhdlmatches {<ref-logic-value>=<test-logic-value>:....}**

(optional) Specifies how to match VHDL signal states in the reference dataset to 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 to match Verilog signal states in the reference dataset to 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 the final argument to the compare configure command.

Related Topics

[compare add](#)

[Waveform Compare \[Questa SIM User's Manual\]](#)

compare continue

Continues comparison difference computations that were suspended with 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 \[Questa SIM User's Manual\]](#)

compare delete

Deletes a comparison object from the currently open comparison.

Syntax

`compare delete [-recursive] {<objectPath>}`

Arguments

- `-recursive`
(optional) Deletes a region recursively.
- `{<objectPath>}`
(required) Path to the comparison object to delete (for example, `{compare:/top\clk<>clk\}`). The comparison object must be "escaped" correctly, so the braces '{ }' and trailing space are required. Must be the final argument to the compare delete command.

Related Topics

[compare add](#)

[Waveform Compare \[Questa SIM User's Manual\]](#)

compare end

Closes the active comparison without saving any information.

Syntax

`compare end`

Arguments

None

Related Topics

[compare add](#)

[Waveform Compare \[Questa SIM User's Manual\]](#)

compare info

Lists the results of the comparison in the Main window transcript. Includes a -write argument to save the information to a file.

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. The default is to not list ignored differences in the output.
- **-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.

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 \[Questa SIM User's Manual\]](#)

compare list

Displays a list in the Transcript window 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 \[Questa SIM User's Manual\]](#)

compare options

Sets defaults for various waveform comparison commands. The defaults are used when other compare commands are invoked during the current session.

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 to automatically add new comparison objects to the Wave window.
(default) You can use the -noaddwave argument to specify not to automatically add objects .
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 to ignore Verilog net strengths 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 display 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 the 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 the 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 not to automatically add new comparison objects to the Wave window. The default is to add comparison objects automatically. Related Tcl variable is PrefCompare(defaultAddToWave).
- **-noignoreVlogStrengths**
(optional) Specifies not to ignore Verilog net strengths 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 synthesis tool bit blasts a bus (breaks it into bits), 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 use 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 to allow the test signal edge to trail or lead the reference edge in an asynchronous comparison.

You can use `-tolLead` and `-tolTrail` to specify different values for the leading and trailing tolerances.

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

`<unit>` — (optional) A suffix specifying a unit of time. The default is to specify the current simulation resolution by omitting `<unit>`. If you specify `<unit>`, you must enclose `<time>` and `<unit>` within curly braces `({})`. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

- `-tolLead <time>[<unit>]`

(optional) Specifies the default maximum amount of time to allow the test signal edge 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>`. If `<unit>` is specified, you must enclose `<time>` and `<unit>` within curly braces `({})`. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

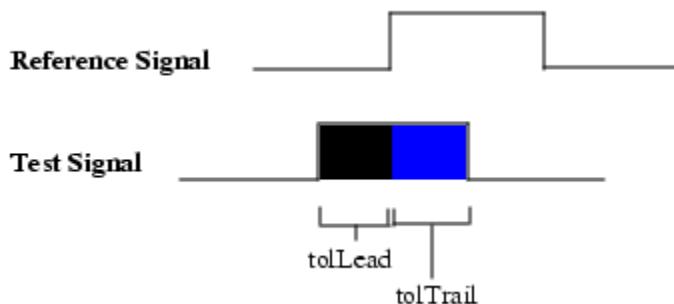
- `-tolTrail <time>[<unit>]`

(optional) Specifies the default maximum amount of time to allow the test signal edge 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>`. If `<unit>` is specified, you must enclose `<time>` and `<unit>` within curly braces `({})`. Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

Figure 2-2. Graphical representation of tolLead and tolTrail



- `-track`

(optional) Specifies to have the waveform comparison track the current simulation. (default) The differences are updated at the end of each `run` command. 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 to match VHDL signal states in the reference dataset to values in the test dataset. Values are specified in a colon-separated list of match values. For example:

```
-vhdlmatches {X=XUD:Z=ZD:L=1HD}
```

Default is:

```
{U=UWX-:X=UWXD:O=OLD:L=1HD:Z=ZD:W=UWXD:L=OLD:H=1HD:-=UX01ZWLHD}
```

Related Tcl variable is PrefCompare(defaultVHDLMatches).

- **-vlogmatches {<ref-logic-value>=<test-logic-value>:...}**

(optional) Specifies how to match Verilog signal states in the reference dataset to values in the test dataset. Values are specified in a colon-separated list of match values. For example:

```
-vlogmatches {O=0:L=1:Z=Z}
```

Default is:

```
{O=0:L=1:Z=Z:X=X}
```

Related Tcl variable is PrefCompare(defaultVLOGMatches).

- **-wavepane <n>**

(optional) Specifies the pane of the Wave window in which to place compare differences.

<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 to add compare differences.

Description

If you do not enter any arguments, compare options returns the current setting for all options. If an option is given that requires a value, and if that value is not given, compare options returns the current value of that option. You can permanently change the default settings in Preferences dialog box (Tools > Edit Preferences), on the By Name tab, under the Compare group of items.

Refer to “Setting GUI Preferences” for details.

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 \[Questa SIM User's Manual\]](#)

compare reload

Reloads comparison differences for viewing without re-computation.

Syntax

compare reload <rulesFilename> <diffsFilename>

Arguments

- <rulesFilename>
(required) Specifies the name of a 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 a file that was previously saved using the compare savediffs command.

Description

Prior to invoking compare reload, you must open the relevant datasets, using the same names that were used during the original comparison.

Related Topics

[compare add](#)
[compare run](#)
[compare savediffs](#)
[compare saverules](#)
[compare start](#)
[Waveform Compare \[Questa SIM User's Manual\]](#)

compare reset

Clears the current compare differences, allowing execution of another compare run command. Does not modify any of the compare options or any of the signals selected for comparison. This enables 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 \[Questa SIM User's Manual\]](#)

compare run

Runs the difference computation on signals you select 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). You can change the simulation resolution with the -t argument of the [vsim](#) command).

<unit> — (optional) A suffix specifying a unit of time. The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit>, you must enclose <startTime> and <unit> within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

- <endTime>[<unit>]

(optional) Specifies when to end computing differences. Can be any positive integer. The default is the end of the dataset simulation run that ends earliest.

<unit> — (optional) A suffix specifying a unit of time. |The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit>, you must enclose <endTime> and <unit> within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

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 \[Questa SIM User's Manual\]](#)

compare savediffs

Saves the comparison results to a file for later reloading.

Note

 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 \[Questa SIM User's Manual\]](#)

compare saverules

Saves the comparison setup information (or "rules") to a file you can re-execute later. Saves compare options, clock definitions, and region and signal selections.

Syntax

`compare saverules [-expand] <rulesFilename>`

Arguments

- `-expand`

(optional) Expands groups specified by the compare add command to individual signals. If you used the compare add command to add a region, and then deleted signals from that region, you must use the `-expand` argument for the rules to reflect the signal deletions.

- `<rulesFilename>`

(required) Specifies the name of the file to which to save the rules. To load the file at a later time, use the [compare reload](#) command. Must be the final argument to the compare saverules command.

Related Topics

[compare add](#)

[compare reload](#)

[compare savediffs](#)

[Waveform Compare \[Questa SIM User's Manual\]](#)

compare see

Displays the specified comparison difference in the Wave window, using any necessary horizontal and vertical scrolling. Highlights the signal containing the specified difference, and positions the active cursor 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 to place the differences.

<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 to add objects.

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 \[Questa SIM User's Manual\]](#)

compare start

Begins a new dataset comparison. The datasets to compare must already be open.

Syntax

```
compare start [-batch] [-maxsignal <n>] [-maxtotal <n>] [-refDelay <delay>[<unit>] ]  
[-testDelay <delay>[<unit>]] [<reference_dataset>] <test_dataset>
```

Arguments

- **-batch**
(optional) Specifies not to automatically insert comparisons into the Wave window.
- **-maxsignal <n>**
(optional) Specifies an upper limit for the total differences encountered on any one signal. When the limit is reached, Questa SIM stops computing differences on that signal. You can change the default with the [compare options](#) command, or by editing the PrefCompare(defaultMaxSignalErrors) variable in the *pref.tcl* file.

 <n> — Any integer. The default is 100.
- **-maxtotal <n>**
(optional) Specifies an upper limit for the total differences encountered. When Questa SIM reaches the limit, it stops computing differences. You can change the default with the [compare options](#) command, or by editing the PrefCompare(defaultMaxTotalErrors) variable in the *pref.tcl* file.

 <n> — Any integer. The default is 100.
- **-refDelay <delay>[<unit>]**
(optional) Delays the reference dataset relative to the test dataset. The simulator applies delays to comparison reference signals you create with the [compare add](#) command, appending the signal name with "_d". The Wave window displays these created signals as comparison objects. The delay does not apply to signals you create when using the -when argument to the compare add command.

 <delay> — Any non-negative integer.

 <unit> — (optional) A suffix specifying a unit of time. The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit>, you must enclose <delay> and <unit> within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.
- **-testDelay <delay>[<unit>]**
(optional) Delays the test dataset relative to the reference dataset. The simulator applies delays to comparison test signals you create with the [compare add](#) command, appending the signal name with "_d". The Wave window displays these created signals as comparison objects. The delay does not apply to signals you create when using the -when argument to the compare add command.

<delay> — Any non-negative integer.

<unit> — (optional) A suffix specifying a unit of time. The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit> is specified, you must enclose <delay> and <unit> within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

- <reference_dataset>

(required) The dataset to use as the comparison reference. Must be the final argument to the compare start command, unless you also specify the <test_dataset> argument.

- <test_dataset>

(optional) The dataset to test against the reference. If not specified, Questa SIM uses the current simulation. The reference and test datasets can be the same.

Examples

- Begin a waveform comparison between a dataset named "gold" and the current simulation. Assumes the gold dataset is already open.

compare start gold

- Open two datasets and start 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 \[Questa SIM User's Manual\]](#)

compare stop

Used internally by the compare stop button to suspend comparison computations in progress.

Syntax

compare stop

Arguments

None

Description

Has no effect on compare runs that have already returned to the VSIM prompt. Under Unix, entering Control-C in the window that invoked Questa SIM has the same effect as compare stop.

Related Topics

[compare run](#)

[compare start](#)

[Waveform Compare \[Questa SIM User's Manual\]](#)

compare update

Primarily used internally to update the comparison differences when comparing a live simulation against a .wlf file.

Syntax

compare update

Arguments

None

Description

Called automatically at the completion of each simulation run if the compare options -track is in effect. You 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 is not in effect when the run command is executed.

Related Topics

[compare run](#)

[compare options](#)

[Waveform Compare \[Questa SIM User's Manual\]](#)

configure

Invokes the List or Wave widget configure command for the current default List or Wave window.

Syntax

Base Command Usage

```
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>]]] [-namecewidth [<width>]] [-rowmargin [<pixels>]]  
[-signalnamewidth [<value>]] [-timecolor [<color>]] [-timeline [0 | 1]]  
[-timelineunits [fs | ps | ns | us | ms | sec | min | hr]] [-valuecewidth [<width>]]  
[-vectorcolor [<color>]] [-waveselectcolor [<color>]] [-waveselectenable [0 | 1]]
```

Description

The command works in three modes:

- With no 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 includes 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

Note

 This list of arguments is not exhaustive. To get the full list of arguments, execute either of the following commands: `lecho [configure list]`, or `lecho [configure wave]`.

- list | wave

(required) Controls the widget to configure. Must be 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 you do not specify a window, 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 ** — (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 displays 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 length of time gating remains 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 is displayed after the gate expression becomes false (if you do not 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 `-gateexpr` argument to specify the expression.) Refer to “[Using Gating Expressions to Control Triggering](#)” in the GUI Reference Manual 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. The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit>, you must enclose <delay> and <unit> within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

- **-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. The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit>, you must enclose <delay> and <unit> within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

- **-usesignaltriggers [0 | 1]**

(optional) Specifies whether or not to use signals as triggers.

0 — Signals are not used as triggers

1 — Signals are used as triggers

- **-usestrobe [0 | 1]**

(optional) Specifies whether or not to use a strobe 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. The default is 2.

- **-cursorlockcolor [<color>]**
 (optional) Specifies the color of a locked cursor. Related Tcl variable is PrefWave(cursorLockColor).
 <color> — Any Tcl color. The default is red.
- **-gridauto [off | on]**
 (optional) Controls the grid period when in simulation time mode.
 off — (default) Uses a user-specified grid period.
 on — The major tick marks in the time line determine the grid period.
- **-gridcolor [<color>]**
 (optional) Specifies the background grid color. Related Tcl variable is PrefWave(gridColor).
 <color> — Any color. The default is grey50.
- **-griddelta [<pixels>]**
 (optional) Specifies how close (in pixels) two grid lines can be drawn before intermediate lines are removed. Related Tcl variable is PrefWave(gridDelta).
 <pixels> — Any non-negative integer. 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. The default is 0.
 <unit> — (optional) A suffix specifying a unit of time. The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit>, you must enclose <delay> and <unit> within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.
- **-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. The default is 1.
 <unit> — (optional) A suffix specifying a unit of time. The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit>, you must enclose <delay> and <unit> within curly braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.
- **-namecolwidth [<width>]**
 (optional) Specifies the width of the name column in pixels. Related Tcl variable is PrefWave(nameColWidth).
 <width> — Any non-negative integer. 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. 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. 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. 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) Display simulation time.
 1 — Display grid period count.
- **-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. The default is 100.

-
- **-vectorcolor [<color>]**
(optional) Specifies the vector waveform color. Default is #b3ffb3. Related Tcl variable is PrefWave(vectorColor).
 <color> — Any color. 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. 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.

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

[Setting GUI Preferences \[Questa SIM GUI Reference Manual\]](#)

context

Provides operations to return information, and in some cases manipulate, a context's name. The argument you specify determines the operation.

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 ‘celldesign 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 is 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>. Discards any trailing path separator.

<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

Displays test information based on a merged UCDB created by a vcover merge operation.

Note

 Some arguments for this command require that you perform the merge to create the data with vcover merge -testassociated, to provide data that is not available by default.

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]`

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: Has an effect only with multiple -select options (does not work with -select instance). These switches are mutually exclusive and you can use only one on each command line.) Use any select criteria (logical OR) or all select criteria (logical AND) to determine what the report displays. 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"
- **-ne** is "not equal to"

The **<value>** is the date you are selecting (or filtering out). The formats accepted for **<value>** are:

- mm/dd[/yy]
- monthname dd[, yy]

- dd monthname [yy]
- [yy]yyymmdd (for example, trend date)
- [yy]yy-mm-dd
- dd-monthname-[yy]yy

In the formats where the year specification is optional, 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, where <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, such as 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 details.

- **-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 by using -aggregate preselected|postselected.

Description

This command is available only 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, while -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 have a '+' sign appended to the node name.

Examples

- Display all coverage items in a specified path:

```
coverage analyze -path /top/*
```

You can use variations of this command to find the path for any specific coverage object in /top, such as a statement.

- List tests that 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 that hit a specific coverage point:

```
coverage analyze -path /top/dut/u1/cvg_inst/cp1 -coverage nonzero
```

- List tests that hit a specific coverage bin:

```
coverage analyze -path /top/dut/u1/cvg_inst/cp1/bin1 -coverage nonzero
```

- List tests that 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:
- Look for any test that 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
```

where the asterisk (*) means “match any number of the preceding characters (i.e 'g' or 'gggg')” and the period (.) means “match any single character”]

Related Topics

[Code Coverage \[Questa SIM User's Manual\]](#)

[Calculation of Total Coverage \[Questa SIM User's Manual\]](#)

[Coverage and Verification Management in the UCDB \[Questa SIM User's Manual\]](#)

[Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)

[Viewing Test Data in the Tracker Window \[Questa Verification Management User's Manual\]](#)

[Verification Management Overview \[Questa Verification Management User's Manual\]](#)

[coverage attribute](#)

[coverage goal](#)

[coverage report](#)

[coverage save](#)

[coverage tag](#)

[coverage unlinked](#)

[coverage weight](#)

coverage attribute

Used to display, set, or delete object, test, and UCDB attributes in the currently loaded database.

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>
```

Description

- 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 “[Predefined Attribute Data](#)” in the User’s Manual for a complete list of these attributes.
- UCDB Attributes — Attached globally to the UCDB file, and 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 Questa SIM creates 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.

You can use this command during simulation or with “vsim -viewcov”, though in simulation you can use it only for test attributes (the single test attribute record that exists in simulation).

To apply filters (-select instance, -assert, -code, and so forth.):

- Match paths first, with recursion (if specified).
- Specify paths to be “thrown out” (those not matching the filter).

You can also use this command to create global user attributes and enable them to capture trend data using the -trendable argument.

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, displays the commands used.
- **-comment [<str>]**
(optional) Comment on the testcase. If <str> is specified, sets a comment; if not, displays comments.
- **-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 display. With -name and -value, this argument can also 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. You can use wildcards.

- **-name <str> -value <int>**

Required for setting attributes; optional for displaying them. Sets attribute name and value. Adds 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 for each 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](#)” in the User’s Manual. 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 remain unique, preventing data conflicts.

- **-path <obj>**

(optional) Apply to a path in the UCDB. You can use <obj> to specify a dataset other than the current dataset. (Refer to “[Object Name Syntax](#)” for instructions on how to specify a dataset.) If you do not specify a dataset, the current dataset is used. Use only one dataset name per command invocation, or an error will result. Wildcards are acceptable. You can use a relative path in conjunction with the -du switch.

- **-plansection <obj>**

(optional) Apply to a testplan section in the UCDB, as specified by <obj>. Wildcards are acceptable. You can use a relative path in conjunction with -du.

- **-seed [<num>]**

(optional) Random seed of the test run. Without <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](#)” in the Verification Management User’s Manual 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 that are in a UCDB that is 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

Note

 In this example, the MERGEINSTANCES attribute is set to 1 to define coverpoint and cross bins for covergroup type-based coverage. By default, the value of MERGEINSTANCES is 0; which means that no bins are created under covergroup types. The only way to define bins under TYPE coverage is to set the MERGEINSTANCES attribute to 1. See [SystemVerilog 2009 type_option.merge_instances](#) for more information.

- 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 \[Questa Verification Management User's Manual\]](#)

[Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)

[Understanding Stored Test Data in the UCDB \[Questa SIM User's Manual\]](#)

[coverage exclude](#)

[coverage goal](#)

[coverage analyze](#)

[coverage ranktest](#)

[coverage report](#)

[coverage save](#)

[coverage tag](#)

[coverage testnames](#)

[coverage weight](#)

[coverage unlinked](#)

coverage clear

Clears specified types of coverage data from the coverage database.

Syntax

```
coverage clear [-code {b | c | e | f | s | t |}...] [-codeAll] [-cvg] [-assert] [-directive]
    [-du <du_name> | -instance <pathname>] [-excluded] [-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 the specified coverage types: b=branch coverage; c=condition coverage; e=expression coverage; s=statement coverage; t=toggle; f=Finite State Machine coverage. You can specify multiple coverage types with a single argument.
- **-codeAll**
(optional) Specifies that the command applies 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 the design unit to clear of specified types of coverage data. To specify all design units in the current dataset, specify <du_name> as “*”.
- **-excluded**
(optional) Immediately clears all coverage exclusions.
- **-instance <pathname>**
(optional) Clears the specified coverage data for the specified instances; you can specify -instance multiple times.
- **-match <string>**
(optional) Recursively matches the given pattern against the specified coverage types through the entire instance tree. If -duname is specified, it matches against the specified coverage types in the design unit. You can use wildcards. This switch and -recursive are mutually exclusive.

- **-path <obj>+**
(optional) Specifies that the subtrees being cleared are rooted at the specified design node. You can specify multiple objects. You can use <obj> to specify a dataset other than the current dataset. (Refer to “[Object Name Syntax](#)” for instructions on how to specify a dataset.) If you do not specify a dataset, the current dataset is used. Use only one dataset name per command invocation, or an error will result. This switch applies to a sub-hierarchy.
- **-recursive**
(optional) Specifies that the command is applied recursively. By default, the query is restricted to the single object or objects specified in the command. This switch and -match are mutually exclusive.

Description

When entered at the simulation prompt (simulation mode), performing coverage clear on an instance affects the code coverage data of the associated design unit, and performing coverage clear on a design unit affects the associated instances of that design unit.

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, and clearing a design unit has no affect on related instances.

Examples

- Clear all coverage data from the current simulation database (UCDB).

coverage clear

- Clear data for all covergroups and covergroup directives.

coverage clear -cvg -directive

- Clear coverage data from all /top/a.

coverage clear -path /top/a/*

Related Topics

[Code Coverage \[Questa SIM User's Manual\]](#)

[coverage attribute](#)

[coverage exclude](#)

[coverage ranktest](#)

[coverage report](#)

[coverage save](#)

[coverage tag](#)

Commands (A - M)

coverage clear

[coverage analyze](#)

coverage create

Creates covergroups, coverpoints, crosses and bins under a specified parent in an existing UCDB file.

Note

 This command allows you to make changes to your hierarchy. Backups are recommended.

Syntax

Create a covergroup, covergroup instance or coverpoint

```
coverage create (-cvg | -cvginstance | -cvp) <name> -parent <path>
```

Create a cross

```
coverage create -cross <name> -parent <path> -item <cvpname1> -item <cvpname2> [-item <cvpnameN>]
```

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 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, creates a regular covergroup bin.
- **-cvg**
(Required for covergroup creation) Specifies creation of a covergroup. Must specify the <name> and the parent <path> for the covergroup.
- **-cvginstance**
(Required for covergroup instance creation) Specifies creation of 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 specified **<path>** must be absolute, not relative. Each coverage create command can specify multiple items.
- **<name>**
(required) Specifies a name for the specified covergroup, coverpoint, cross, or bin that is unique within the parent scope.
- **-parent <path>**
(required) Specifies the path to the parent item of the specified covergroup, coverpoint, or cross. The specified **<path>** must be absolute.

Description

You can use coverage create only in Coverage View mode (**vsim -viewcov...**). Once created, the covergroup items behave like any SystemVerilog covergroup items, and coverage calculation follows SystemVerilog coverage calculation rules.

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
```

Note

 In this example, the **MERGEINSTANCES** attribute is set to 1 in order to define coverpoint and cross bins for covergroup type-based coverage. By default, the value of **MERGEINSTANCES** is 0; which means that no bins are created under covergroup types. The only way to define bins under TYPE coverage is to set the **MERGEINSTANCES** attribute to 1. See [SystemVerilog 2009](#) **type_option.merge_instances** for more information.

- 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 i1 -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 \[Questa SIM User's Manual\]](#)

[coverage attribute](#)

[coverage exclude](#)

[coverage ranktest](#)

[coverage report](#)

[coverage save](#)

[coverage tag](#)

[coverage analyze](#)

coverage edit

Opens a coverage dataset (*.ucdb*) to edit the contents. Used only in Coverage View mode (vsim -viewcov...). Use this command to alter existing UCDBs.

Caution

 This command allows you to make changes to your hierarchy. Be aware that some changes in hierarchy can render your data unusable by the tools. Backups are recommended.

Syntax

Prune objects from the UCDB

```
coverage edit -keeponly | -delete {-path <path> | -plansection <path> | -instofdu <du_name> | -test <test_name><coverage_types>]...  
<coverage_types> = ( [-code {b | c | e | f | s | t}...] [-codeAll] [-cvg] [-directive] [-assert] )
```

Change or move around levels/hierarchies within the UCDB

```
coverage edit -stripdesign <int> | -installdesign <path> ...  
coverage edit -stripplan <int> | -installplan <path> ...
```

Move instances of a design or trees/subtrees/instances of a testplan

```
coverage edit -movedesign <source path> <dest path>  
coverage edit -moveplan <source path> <dest path>
```

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> }
```

Rename libraries or sub-sections of libraries

```
coverage edit -rename <new_name> -lib <old_libname> [-path <path> | -instofdu <du_name> ]
```

Edit paths for HDL source code

```
coverage edit -rename -srcfilestring <target_substring> <replacement_substring>
```

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. You can specify multiple coverage types in 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 the specified paths.

Note

 This command does not support deleting a test from a merge file. Any attempt to delete a test from a merge file using ‘coverage edit -delete -test <test_name>’ generates an error message.

- **-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 specified `<path>` 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>` must be absolute, not relative. You can use this argument multiple times in a coverage edit command.
- **-installplan <path>**
(optional) Creates the specified new testplan scope(s) to the testplan tree, at the specified `<path>`. The `<path>` must be absolute, not relative. You can use this argument multiple times in a 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 the specified paths.

- **-lib <old_libname>**
(optional) Valid only for use with the -rename argument. Specifies a particular library instance name to replace 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 the specified paths.
- **-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 the specified paths.
- **-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 the full path, but only the scope to rename (accepts wildcards). To identify the hierarchical path to the target scope to rename, use 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 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. Searches the <target_substring>; the <replacement_substring> replaces the first matching string found in the path to the VHDL or Verilog source. Does not support wildcards. You can merge the resulting UCDB 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. Stripping a design also removes non-instance subscopes of that design. If specified levels are out of range, a warning is issued. You can specify -stripdesign multiple times in a coverage edit command.

- **-stripplan <int>**
(optional) Deletes the specified number of levels (<int>) of topmost testplan scopes in the testplan tree. If specified levels are out of range, a warning is issued. You can specify -stripplan multiple times in a coverage edit command.
- **-test <test_name>**
Edits a specified test name. Mutually exclusive with -path, -plan, and -instofdu.

Description

The strip and install versions of the command are useful for merging a unit test module and a system test module.

Examples

Note

 Coverage View mode does not support the use of spaces and brackets ([] in arguments.

Some of the examples below show the use of braces ({} as escape characters around arguments that includes those characters, and around null arguments.

- 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* (note the use of braces ({} as escape characters):
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" (note the use of braces ({})) to identify a null character):

coverage edit –rename –lib {} libA

- Remove the library name part of "work" design units (note the use of braces ({})) to identify a null character):

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 \[Questa SIM User's Manual\]](#)

[Coverage and Verification Management in the UCDB \[Questa SIM User's Manual\]](#)

[Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)

[coverage analyze](#)

[coverage attribute](#)

[coverage exclude](#)

[coverage ranktest](#)

[coverage report](#)

[coverage save](#)

[coverage tag](#)

[coverage weight](#)

coverage exclude

Enables you to exclude items from coverage statistics.

Syntax

Exclude comment-related commands

```
coverage exclude <exclusion> -comment <str>
```

Note

 This usage is available only in Coverage View mode.

File-based, line-based, or wholesale exclusions

```
coverage exclude {-srcfile <source_file_list> [-pragma] | -du <du_name> [-srcfile  
    <source_file_list>] [-pragma] | -scope <path_list> [-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] [-reason <str>]
```

Exclude expression or condition rows

```
coverage exclude {-srcfile <source_file_list> [-pragma] | -du <du_name> [-srcfile  
    <source_file_list>] [-pragma] | -scope <path_list> [-srcfile <source_file_list>]} [-item  
    {<bces>} [[<int> | <int-int>] ...]] [-feccondrow <ln> [<rn>] ... [<rn>-<rn>] ...] [-fecexprrow  
    <ln> [<rn>] ... [<rn>-<rn>] ...] [-dataset <name>] [-clear]
```

Exclude functional coverage

```
coverage exclude {-assertpath <assert_list> | -dirpath <directive_list>}  
    [-du <du_name> | -srcfile <source_file_list>] [-clear] [-comment <str>]
```

Note

 Allows wildcard characters when running the command in Coverage View mode.

Exclude assertions and/or cover directives by line range

```
coverage exclude { -srcfile <source_file_list> | -du <du_name> [-srcfile <source_file_list> |  
    -scope <path_list> [-srcfile <source_file_list> | -recursive]} [-linerange [<ln>] ... [<ln>-  
    <ln>] ...] [-comment <str>] [-code {a | b | c | d | e | f | s | t}...] [-immediate] [-concurrent] [-  
    dataset <name>] [-clear] [-lang sv | psl | vhdl] [-severity note | warning | error | fatal]
```

Note

 Immediate and concurrent assertions/cover directives are specified with the -immediate and
-concurrent options, respectively. If you do not specify either -immediate nor -concurrent,
both types are excluded.

If you do not specify -code, the exclusion applies to assertions and cover directives as well as
code coverage types (equivalent to -code abcdefst.)

Exclude FSM states or state transitions

```
coverage exclude (-du <du_name> {<fsm_args> | -pragma | -scope <path_list>} | [-dataset <name>]) [-clear]
<fsm_args> = -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]
```

Exclude toggle coverage

```
coverage exclude -togglenode <node_path_list> [-du <du_name> | -scope <path_list>
[-recursive]] [-dataset <name>] [-in] [-out] [-inout] [-internal] [-ports] [-clear] [-pragma]
[-trans <transition_list>]
```

Exclude power states and transitions (Power Aware simulation)

```
coverage exclude -scope <path_list> {-pstate <state_var_name> [<state_list>]} |
{-ptrans <state_var_name> [<transition_list>]} [-pcross] [-clear] [-comment <str>]
[-reason <str>]
```

Description

The coverage exclude command enables 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)
- power states and transitions

You can issue this command and its arguments (except -comment) during simulation, or in Coverage View (post-process) mode.

Refer to “[Methods for Excluding Objects](#)” and “[Excluding Functional Coverage from the GUI and Reports](#)” in the User’s Manual for details on the different methods for excluding code and functional coverage, and a description of the advantages of particular methods.

Use the -comment option to add/edit comments for an exclusion (supported for use in Coverage View mode only).

Use the -clear switch to remove Coverage exclusions.

You can use wildcard characters and shortcuts. See specific arguments for usage information and the Examples section for example commands.

You cannot clear file based exclusions by scope. For example, an exclusion that was set using -srcfile cannot be cleared later using -scope.

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](#)” and “[Exclude AllFalse Branches in Case Statements](#)” in the User’s Manual.
- **-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. You can specify more than one coverage with each -code argument. If -item is used, -code f or t is not valid.
- **-clear**
(optional) Removes exclusions from the coverage metrics. The “coverage exclude” command adds exclusions whenever -clear is not specified.
- **-comment <str>**
(optional) Attaches a comment string describing the exclusion to the excluded item. Valid in post-processing (Coverage View) mode only. The comment appears in the coverage exclusions reports created by coverage report and vcover report. You can see these comments by hovering the cursor over the line in the Hits column of the HTML reports and

the Source window. A plus sign, '+', is added in the HTML cell to indicate that this cell has an exclusion comment. Clearing an exclusion clears any associated comment. You can add, modify and remove comments either from the GUI, or by reapplying the same command used initially, replacing the <string> with a new comment. Using an empty <string> for the comment removes the comment.

- **-concurrent**
(optional) Excludes concurrent assertions/cover directives only.
- **-cvgpath <cvg_list>**
(optional) Excludes the covergroup with the specified path from coverage. Allows wildcard usage in Coverage View mode only.
- **-dataset <name>**
(optional) Specifies dataset in which to apply exclusions. Use only one dataset name per command invocation, or an error results. 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. Allows wildcard usage in Coverage View mode only.
- **-du <du_name>**
(required only for du-based exclusions) Specifies design unit to exclude. Allows multiple -du specifications. Mutually exclusive with -scope, which is the default. The <du_name> must include the path to the design unit to exclude. Supports wildcards. For example, you can specify all design units in the current dataset with the "*" wildcard – for example, coverage exclude -du *.
- **-fcontains <state_var_name> [<two_state_trans_list>]**
(optional; must be used with -du fsm) Excludes 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.
- **-fecondrow <ln> [<rn>] ... [<rn>-<rn>] ...**
(optional) Excludes specified row in focused expression coverage (FEC) condition coverage with a specified line number from the report. Allows multiple rows, or ranges of rows, separated by spaces. If you do not specify a row number, all rows are excluded. You can use "-fcr" as a shortcut for this argument.
- **-fecexprrow <ln> [<rn>] ... [<rn>-<rn>] ...**
(optional) Excludes specified row in focused expression coverage (FEC) expression coverage with a specified line number from the report. Allows multiple rows, or ranges of

rows, separated by spaces. If you do not specify a row number, all rows are excluded. You may use “-fer” as a shortcut for this argument.

- **-finitial <state_var_name> [<state_list>]**

(optional; must be used with -du fsm) Excludes 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 applies.

- **-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 applies.

- **-fstate <state_var_name> [<state_name>] ...**

(optional) Specifies the Finite State Machine state or states to exclude from coverage for the specified FSM, specified with <state_var_name>. Allows multiple states, separated by white space. If you do not specify a state name, 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 **AutoExclusionsDisable** variable in the *modelsim.ini* file. (Refer to [AutoExclusionsDisable](#) in the User’s Manual.)

- **-ftrans <state_var_name> [<transition_name>] ...**

(optional) Specifies the transition states to exclude for the specified FSM (state_var_name). <transition_name> is "<state_name>-><state_name>". Multiple transitions, separated by white space, are allowed. If you do not specify a transition, all transitions are excluded. If whitespace is present within the transition, you must enclose it in 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 be applied only to coverage exclude command entries for the line number specified with **-linerange**. **<bces>** is required and specifies one or more of the coverage types to exclude: branch, condition, expression, and/or statement. Items are numbered from left to right within a line, regardless of hierarchy, from 1 upward. Only one **-item** argument is allowed with each coverage exclude command. You cannot use this argument with the **-code tf** argument.
- **-lang sv | psl | vhdl**
(optional) Specifies assertions/cover directives of a specific language (SV, PSL, or VHDL). You can specify multiple languages by using this option multiple times. If you do not specify **-lang**, all three languages are selected.
- **-linerange [<ln>] ... [<ln>-<ln>] ...**
(optional) Specifies the line number(s) and/or range of line numbers to exclude from code coverage in the design source file **-srcfile <source_file>**. Allows multiple lines and line ranges, separated by whitespace.
 - Select multi-line statements, branches, conditions or expressions by specifying the last line number of that item.
 - If **-linerange** is not specified, all objects on all lines of the specified design unit, scope, or source file are excluded (a “wholesale exclusion”.)
 - **-srcfile** is required for **-linerange**, unless you use **-du** or **-scope**, and use only one source file to implement the **du** or **scope**.
 - If **-srcfile** is used 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 you specify **-togglenode**.
- **-ports**
(optional) Excludes the specified toggle nodes of mode IN, OUT, or INOUT. This argument is valid only when you specify **-togglenode**.
- **-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.

- **-pstate <state_var_name> [<state_list>]**

Excludes the states listed in *<state_list>* of the object *<state_var_name>* from the coverage metrics. Use the argument with the [-scope <path_list>](#) argument.

Note

 It also excludes all transitions to and from the specified state.

<state_var_name> — Name of any of the following objects:

- Power domain
- Supply port
- Power state table
- Supply set
- Power switch
- Power switch control and ack port
- Isolation enable signal
- Retention save or restore signal

<state_list> — Space-separated list of states of the object *<state_var_name>* that are to be excluded from the coverage metrics.

- **-ptrans <state_var_name> [<transition_list>]**

Excludes the transitions listed in *<transition_list>* of the object *<state_var_name>* from the coverage metrics. Use the argument with the [-scope <path_list>](#) argument.

<state_var_name> — Name of any of the following objects:

- Power domain
- Supply port
- Power state table
- Supply set
- Power switch
- Power switch control and ack port
- Isolation enable signal
- Retention save or restore signal

<transition_list> — Space-separated list of transitions of the object *<state_var_name>* that are to be excluded from the coverage metrics. Wildcard character (*) is allowed.

Syntax:

```
state_value1 -> state_value2
* -> state_value3
state_value1 -> *
```

- **-pcross**

Excludes cross coverage bins of the states and transitions listed with the **-pstate** and **-ptrans** argument.

- **-reason <str>**

(optional) Enables you to specify exclusion reasons. Valid values are the codes given in the [Auto Exclusions](#) section of the User's manual.

- **-recursive**

(optional) Used with the **-scope** argument only. Specifies that exclusions apply recursively into subscopes. If you omit this argument, exclusions are limited to the current scope.

- **-severity note | warning | error | fatal**

(optional) Specifies the assertion/cover directive severity level. Specifying **-severity** selects only assertions/cover directives with the same or lower severity. If not specified, assertions/cover directives of all severities are selected.

- **-scope <path_list>**

(required only for instance-based exclusions) Specifies the scope to exclude. Mutually exclusive with the **-du** argument; **-scope** is the default if neither argument is used. Allows multiple **-scope** specifications. Does not allow wildcards. To recursively exclude scopes, use with **-recursive**.

- **-srcfile <source_file_list>**

(required only for file-based exclusions) Specifies source file to exclude. Allows multiple file specifications, separated by white space.

- **-togglenode <node_path_list>**

(optional) Specifies the named nodes for toggle exclusion. Allows multiple nodes separated by spaces. Part-selects are accepted — with node and selects surrounded by 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, the specified toggle nodes to exclude are relative to the current scope. If used with **-du**, specified toggle nodes to exclude are relative to the design unit.

- **-trans <transition_list>**

(optional) Specifies the toggle node transition(s) to exclude from coverage. Allows multiple transitions separated by spaces. Transition names are not case sensitive and can be any of the following six: 01 10 0Z 1Z Z0 Z1

- **-transitive**

(optional) When a coverpoint or cross bin is excluded, all associated cross bins from all crosses where the coverpoint participates directly or hierarchically (cross of crosses) are also excluded, if the **-transitive** argument is used with the **-cvxpath <cvxpath>** argument. Do not use **-transitive** with the **-clear** argument.

Examples

- Recursively exclude branch coverage from instance */top/dut*.

```
coverage exclude -scope /top/dut -recursive -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
```

- 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> -cvxpath *
```

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 -recursive

- 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

- allfalse has no effect because branch coverage is not specified.

coverage exclude -srcfile project1.vhd -code s -allfalse

- There is no file name and line number associated with FSM and toggle coverage.

coverage exclude -srcfile project1.vhd -linerange 10-20 -code ft

- recursive does not work with -srcfile or -linerange

coverage exclude -scope /top/dut -srcfile project1.vhd -linerange 10-20 -recursive

- '-du *' does not work with -srcfile or -linerange

coverage exclude -du * -srcfile project1.vhd -linerange 10-20

- -pragma does not work with -scope for FSM coverage

coverage exclude -scope /top/dut -srcfile project1.vhd -line 10-20 -pragma

Power Aware cross coverage exclusion

- Exclude a state SLEEP in the power domain PD_SYS1 in instance */top/pd_alu*.

coverage exclude -scope /top/pd_alu -pstate PD_SYS1 SLEEP
- Exclude all transitions to the state SLP in the power domain PD_SYS2 in instance */top/pd_alu*.

coverage exclude -scope /top/pd_alu -ptrans PD_SYS2 * -> SLP
- Clear all exclusions to the state SLP in the power domain PD_SYS2 in instance */top/pd_alu*.

coverage exclude -scope /top/pd_alu -ptrans PD_SYS2 * -> SLP -clear
- Exclude all cross coverage bins of the state SLEEP in the power domain PD_SYS1 in instance */top/pd_alu*. This will not exclude the state and transition coverage.

coverage exclude -scope /top/pd_alu -pstate PD_SYS1 SLEEP -pcross

Related Topics

[Code Coverage \[Questa SIM User's Manual\]](#)

[Coverage Exclusions \[Questa SIM User's Manual\]](#)

[Coverage and Verification Management in the UCDB \[Questa SIM User's Manual\]](#)

[Verification Management Browser Window \[Questa SIM GUI Reference Manual\]](#)

[coverage analyze](#)

[coverage report](#)

[coverage save](#)

[“Toggle Coverage \[Questa SIM User's Manual\]](#)

coverage goal

Sets or displays the values for goals that affect bins, coverage items and specific scopes in the design hierarchy.

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**] **<goal_number>**

To display a currently set goal for any of the above, specify the command without **<goal_number>**.

Description

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 and 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.

To display goals, enter a coverage goal command without an **<integer_weight>**.

For more information regarding the gathering of coverage statistics, refer to “[Calculation of Total Coverage](#)” in the User’s Manual.

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 no other assertion sub-types 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. You must specify at least one coverage type with each -code argument, and you can specify more than one type.
- -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. You can use <path> to specify a dataset other than the current dataset. If no dataset is specified, the current dataset is used. Use only one dataset name per command invocation, or you will get an error. 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.
- **-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 \[Questa SIM User's Manual\]](#)

[Coverage and Verification Management in the UCDB \[Questa SIM User's Manual\]](#)

[Verification Management Overview \[Questa Verification Management User's Manual\]](#)

[coverage analyze](#)

[coverage attribute](#)

[coverage exclude](#)

[coverage ranktest](#)

[coverage report](#)

[coverage save](#)

[coverage tag](#)

[coverage weight](#)

coverage loadtestassoc

Loads test associated data in preparation for using any test association procedures.

Syntax

`coverage loadtestassoc`

Arguments

None

Description

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).

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 \[Questa SIM User's Manual\]](#)

[Coverage and Verification Management in the UCDB \[Questa SIM User's Manual\]](#)

[Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)

[coverage analyze](#)

[coverage attribute](#)

[coverage ranktest](#)

[coverage tag](#)

[coverage weight](#)

coverage open

Opens UCDB datasets for viewing in the GUI in Coverage View mode.

Syntax

coverage open <filename> [<logicalname>]

Arguments

- <filename>
(required) Specifies the <filename>.ucdb to open in Coverage View mode. You must enter at least one UCDB.
- <logicalname>
(optional) Specifies the logical name for the UCDB dataset. This is a prefix that identifies the dataset in the current session. By default the dataset prefix is the name of the specified UCDB file.

Description

This command is equivalent to the command vsim -viewcov. You can close open datasets with [dataset close](#).

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 \[Questa SIM User's Manual\]](#)

[Coverage and Verification Management in the UCDB \[Questa SIM User's Manual\]](#)

[Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)

[coverage analyze](#)

[coverage attribute](#)

[coverage exclude](#)

[coverage ranktest](#)

[coverage report](#)

[coverage save](#)

[coverage tag](#)

[coverage weight](#)

coverage ranktest

Ranks coverage data contained in the current Coverage View dataset loaded with vsim -viewcov, according to each individual test's contribution to cumulative coverage.

Syntax

```
coverage ranktest [<ranktest_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>]
  [-j <number_of_parallel_processes>]
  [-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>]
```

Description

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 tests written to *stdout* or to a file you specify (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 covera ge %
-----------------------------	---	--------------------------------------

Table 2-4. Order and Type of Ranked Tests (cont.)

Contributing, noncompulsory	Tests providing coverage not provided by any previous test.	Sorted by total covera ge %
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.

Arguments

- **-algorithm {greedy | quick}**

Specifies the algorithm to use to produce the ranked list of tests: greedy is the default algorithm. It produces the shortest list of tests contributing to coverage, arranged in order from highest to least incremental contribution. The quick algorithm provides an estimated ordering of tests, which may not be the shortest list, and may not have the tests in an order that perfectly determines 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 halts 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 be used only in a mutually exclusive fashion with the other coverage values.

Any given assertion can have both passes and failures. Any given assertion can also possibly never be evaluated to completion. This means that the sum of assertion passes and assertion failures will often 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 halts 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. You can specify more than one coverage type with each -code argument (example: “-code bcest”). This argument is not compatible with -metric total: if both are explicitly set, ranking halts 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 halts 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 do not display in the detailed ranktest listing. In the GUI, the accumulated coverage data and incremental coverage data for compulsory tests displays 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 in 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 in this order. In the GUI, the accumulated coverage data and incremental coverage data for compulsory tests is calculated and displayed. This option can have the worst performance of the three options.
- **-concise**
(optional) Specifies to create the output 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 by 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 halts 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 halts 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 assumes. 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. Use a comma-separated list to specify multiple message numbers.

Edit the variable in the *modelsim.ini* file to set a permanent default. Refer to "[Message Severity Level](#)" in the User's Manual 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.

- **-j <number_of_parallel_processes>**

(optional) Sets 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 halts and an error is issued.

<coverage_type> — Specifies to apply the goal 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.

You can specify the **-goal** argument 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 allow. If the specified number of seconds is exceeded, the ranking process stops.

<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 rank. When this threshold is exceeded, the ranking operation terminates. <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](#)” in the User’s Manual 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. Specify multiple message numbers as a comma separated list.
Edit the variable in the *modelsim.ini* file to set a permanent default. Refer to “[Message Severity Level](#)” in the User’s Manual for more information
- **-path <path>**
Restricts ranking to design paths (non-testplan) matching the specified <path>. Optional. You can use this argument 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 is 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 is 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 of the ranktest file being created. Default is *ranktest.rank*. Can be specified with the [vcover stats](#) command to redisplay the results of this ranking. You can use this file to repopulate the Browser with ranktest information.
 <filename> — A user specified string, which can include a path.
- **-simtime**
(optional) Specifies that tests being selected for ranking are ordered by 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. Specify multiple message numbers as a comma separated list.

Edit the variable in the *modelsim.ini* file to set a permanent default. Refer to “[Message Severity Level](#)” for more information.

- **<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. Specify multiple message numbers as a comma separated list.

Edit the 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 halts and an error is issued.

<coverage_type> — Specifies that the goal applies only to the specified type of coverage; otherwise, the goal applies 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.

You can specify the `-weight` argument multiple times, as shown in this example:

```
coverage ranktest -weight -assertion 10 -weight -code bcest 5
```

Related Topics

[Code Coverage \[Questa SIM User's Manual\]](#)

[Coverage and Verification Management in the UCDB \[Questa SIM User's Manual\]](#)

[Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)

[coverage analyze](#)

[coverage goal](#)

[coverage tag](#)

[coverage unlinked](#)

[coverage weight](#)

coverage report

Produces textual output of coverage statistics or exclusions. By default, the command prints results to the Transcript window and returns the report as a string. Use the `-out <filename>` argument to save the output to a file.

Syntax

```
coverage report [<coverage_arguments>]
```

Coverage Arguments

```
coverage report [-above <percent>] [-below <percent>] [-append] [-binrhs] [-bydu] [-byfile] [-by instance] [-concurrent |-immediate] [-details] [-file <filename>] [-flat [-primarykey <type | path | value>] [-secondarykey <type | path | value>]]] [-memory] [-nocomment] [-noexcludedhits] [-nomissing] [-nozeroweights] [-pa] [-pacombined] [-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 <du_name>] [-hidecvginsts | -hidecvginstspi0] [-instance <path>] [-notimestamps] [-nozeroweights] [-pa] [-pacombined] [-plansection <path>] [-precision <int>] [-showexcluded] [-summary] [-showcvggoalpcnt] [-testhitdata[=<int>]] [-testhitdataAll] [-testdetails] [-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] [x]}...] [-codeAll] [-cvg] [-directive] [-failed] [-filter <pattern>] [-lang sva | psl | vhdl] [-severity info | note | warning | error | failure | fatal] [-testattr]
```

Note

-  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

```
coverage report [-coverenhanced] [-du <du_name>] [-instance <path>] [-library <libname>] [-noannotate] [-package <pkgname>] [-source <filename>]
```

For a Report of Non-collected Coverage Items

```
coverage report -nocollect [-out <filename> [-append]]
```

Exclusion Coverage Arguments

```
coverage report -excluded [-adaptive] [-pragma] [-user] [-nocomment] [-pa] [-code {[b] [c] [e] [f] [s] [t] [x]}...] [-instance <path>] [-file <filename> [-append]] [-showexcluded]
```

Toggle-specific Coverage Arguments

coverage report [-portmode (input | output | inout | internal)] [-verbose] [-all]

Note

 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 `-by instance`, `-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] [-unattempteddimmed]

Cover directive-specific Arguments for SystemVerilog and PSL

coverage report [-config]

Covergroup-specific Arguments

coverage report [-covered] [-cvg] [-cvp] [-hidecvginsts | -hidecvginstspi0] [-nocvbinsummary] [-nocrossbinsummary] [-nocvgbin] [-noignorebins] [-option] [-samples [[-le | -ge] <int> | <int>-<int>]] [-usecnpm] [-uselimit]

Description

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 by right-clicking in the Structure or Files windows and selecting Code Coverage > Coverage Reports from the popup context menu; or from Tools > Code Coverage > Report.

Tip

 HTML reports are created with the `-html` option, which is compatible only 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

i 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."

To produce reports offline (for example, without a simulation loaded), use the [vcover report](#) command.

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. Reports counts of all enumeration values. 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) Adds assertion coverage data to the report.
- **-below <percent>**
(optional) Specifies that the output include only objects with coverage values below this percentage. <percent> is the coverage of coverpoints and crosses, not covergroups. See also -above.
- **-binrhs**
(optional) Specifies that for covergroups, the report includes a column displaying 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 iterates through all design units in the design and reports coverage data for each. Each design unit report is the merged result of all instances of that design unit, 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 with the [-du <du_name>](#) argument.
- **-byfile**
(optional) Writes out a coverage summary for each source file in the design. This is the default report.

- **-by instance**
(optional) Writes out a coverage summary for all instances and packages. Can replace the default (-byfile) with the -setdefault byinstance argument. Used with the **-recursive [-depth <n>]** argument, reports on all instances below the specified instance.
- **-code {[b] [c] [e] [f] [s] [t] [x]}...**
(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 enable at compile time. You can specify more than one coverage type 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.
- x — Include extended 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.
- **-concurrent |-immediate**
(optional) Selects only concurrent assertions when the -concurrent switch is used. Selects only immediate assertions when the -immediate switch is used. These are mutually exclusive options. If you do not specify either switch, both concurrent and immediate assertions are selected.
- **-config**
(optional) Specifies to include the current configuration of each cover directive in the report.
- **-covered**
(optional) Displays covered bins only. This argument applies only to covergroups in coverage reports and the GUI. If any covergroup item survives the covergroup filters applied, its ancestors display in the report and the GUI.
- **-coverenhanced**
(optional) Enables non-critical functionality that might change the appearance or content of coverage metrics. This argument has an effect only in letter releases (such as 10.Xa, 10.Xb,

10.Yc). Major releases (such as 10.X, 10.Y, 10.Z), enable all coverage enhancements present in previous letter release streams 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. To view these details, search the release notes using the string "coverenhanced".

- **-cvg**
(optional) Adds covergroup coverage data to the report.
- **-cvp**
(optional) Adds specified coverpoint or cross data to the report.
- **-details**
(optional) Includes details associated with each coverage item in the output (FEC). By default, details are not provided.
- **-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. Allows multiple -du specifications per coverage report command.
- **-excluded [-adaptive] [-pragma] [-user] [-nocomment] [-pa]**
(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.
 - adaptive — When used with the -excluded argument, generates an adaptive exclusion DO file.
 - 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 that appear by default in the exclusion report created with -excluded. Comments can be added, modified, and removed from the GUI.
 - pa — Lists Power Aware objects that have been excluded.
- **-failed**
(optional) Reports only failed assertions/cover directives.

- **-file <filename>**
(optional) Specifies a file name for the output report. Default is to write the report to the Transcript window. You can use environment variables in the pathname.
- **-filter <pattern>**
(optional) Specifies that the command affects only those objects with names that match the specified pattern. Must be specified with the -assert or -directive argument.
 <pattern> — Specifies the character(s) to match in the search. Permits wildcards.
- **-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 <du_name>] [-hidecvginsts | -hidecvginstspi0] [-instance <path>] [-notimestamps] [-nozeroweights] [-pa] [-pacombined] [-plansection <path>] [-precision <int>] [-showexcluded] [-summary] [-showcvggoalpcnt] [-testhitdata[=<int>]] [-testhitdataAll] [-testdetails] [-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 coverage 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. Allows only one input UCDB.
- 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 <du_name> — (optional) Includes only information for the specified design unit in the HTML report; by default, all design unit information is included. You can use the -du option multiple times in the command to specify more than one instance. You can also use wildcards (*).
- hidecvginsts | -hidecvginstspi0 — (optional) Removes from the HTML report all covergroup instances, or only those having per_instance set to 0.

- instance <path> — (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. You can use the -instance option multiple times in the command to specify more than one instance. You can also use wildcards (*).
- 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 sections. Has no effect on coverage numbers.
- pa — Generates the Power Aware coverage report, which contains information about Power Aware coverage. Use this argument with the -details, -verbose, -assert, -file, -html, and -xml arguments. Refer to “[Generating the Power Aware Coverage Report](#)” in the “*Power Aware Simulation User’s Manual*” for more information.
- pacombined — Generates the basic and advanced combined coverage report, which contains information about Power Aware and non-Power Aware coverage. Use this argument with the -details, -verbose, -assert, -file, -html, and -xml arguments. Refer to “[Generating the Basic Combined Coverage Report](#)” and “[Generating the Advanced Combined Coverage Report](#)” in the *Power Aware Simulation User’s Manual* for more information.
- plansection <path> — specifies particular verification plan sections to which the report is applied.
- precision <int> — (optional) Sets the precision for the values displayed in HTML report.
- showcvgoalpcnt — (optional) Displays a column for Goal % in the covergroups table. If you do not specify this argument, the value of Goal % is displayed in the tooltip for each covergroup.
- showexcluded — (optional) Shows the excluded items in the HTML report; by default excluded items are not shown.
- summary — (optional) Includes only the top summary page, the summary page, and the list of tests run in the generated report.
- testhitdata[=<int>] — (optional) Enables generation of a table containing test-bins-hit information for the first 10 tests that hit a bin. When an integer is specified with [= <int>], the table will include the number of tests that hit a bin as specified by the integer. When the -testhitdata 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.
- testhitdataAll — (optional) Enables generation of a table containing test-bins-hit information for all tests that hit a bin.
- testdetails — (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 of 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. You can use <path> to specify a dataset other than the current dataset. (Refer to “[Object Name Syntax](#)” for instructions on how to specify a dataset.) Uses the current dataset if none is specified. You can use only one dataset name per command invocation, or an error results.

- **-lang sva | psl | vhdl**

(optional) Specifies assertions of a specific language (SVA, PSL, or VHDL). You can use this option multiple times to specify multiple languages. If you do not specify -lang, all three languages are selected.

- **-library <libname>**

(optional) Use 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 -out argument, and optionally, -append. Used with -out to print the output to a specified file name, and -append to append the data to the end of the file.
- **-nocomment**
(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.
- **-nocvbinsummary**
(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) Disables display of "E-hit" notifications. 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. The -noexcludedhits option makes all "E" and "EA" type notifications uniformly visible, along with exclusion reasons (if available).
- **-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.
- **-nozeroweights**
(optional) Prevents any items with zero weights from appearing in the report, including all zero-weighted coverage items and 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.

- **-pa**
Generates a report on coverage that is applied for a Power Aware simulation. You can use this argument with the -assert, -details, -html, -verbose, and -xml arguments. Refer to “[Generating the Power Aware Coverage Report](#)” in the *Power Aware Simulation User’s Manual* for more detailed information on using these arguments for a coverage report.
- **-pacombined**
Generates a combined coverage report for a non-Power Aware and a Power Aware simulation. You can use this argument with the -assert, -details, -html, -verbose, and -xml arguments. Refer to “[Generating the Basic Combined Coverage Report](#)” and “[Generating the Advanced Combined Coverage Report](#)” in the *Power Aware Simulation User’s Manual* for more information.
- **-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 [[-le | -ge] <int> | <int>-<int>]**
(optional) Filters the sample count for covergroup types. You can optionally calculate the sample count with **SVCovergroupsSampleInfo**, set in the *modelsim.ini* file. Refer to [SVCovergroupSampleInfo](#) in the User’s Manual. This argument filters to include covergroups whose sample count matches the criteria given in the argument. It is a filter for covergroup types only, not instances. If covergroups are found without a sample count, a warning is issued. An argument can be one of the following four mutually exclusive options:
 - [int] — sample count is equal to [int]
 - le [int] — sample count is less than or equal to number
 - ge [int] — sample count is greater than or equal to number
 - [int]-[int] — (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 and code coverage items.
- **-source <filename>**
(optional) Writes a summary of statement coverage data for a specific source file. You can use environment variables 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. You can apply multiple -testextract arguments in same command. This argument is compatible with reports generated in plain text and XML formats only; HTML reports are not supported. When you use 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 the report. (Refer to **SVCrossNumPrintMissingDefault** in the User's Manual.) By default, the report displays all cross bins.
- **-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 the default value of SV. 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. The list includes 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](#)” in the User's Manual 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 -instance option.
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 -out myreport.txt
- Write detailed branch, condition, and statement statistics, without associated source code, to the transcript window.
coverage report -details -noannotate -code bcs

- Write code coverage details to *myreport.txt*. The -details argument reports coverage statistics for each statement, branch, condition and expression.

coverage report -details -out 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* including both toggled and untoggled signals.

coverage report -details -testextract clyde40ns -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 -out myreport.txt

- Write a list of statements with zero coverage to *myzerocov.txt*.

coverage report -zeros -out myzerocov.txt

Related Topics

[Code Coverage \[Questa SIM User's Manual\]](#)

[Generating HTML Coverage Reports \[Questa SIM User's Manual\]](#)

[coverage save](#)

[coverage analyze](#)

[coverage attribute](#)

[coverage ranktest](#)

[coverage tag](#)

[coverage goal](#)

[coverage weight](#)

coverage save

Saves the coverage results of the specified type from a simulation to the unified coverage database (UCDB). If no type is specified, saves all types into the database.

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>]
[-noadaptive] [-norecursive] [-onexit] [-seed] [-ovmtestname] [-uvmtestname]
[-seed] [-coveredfcov] [-cvg] [-pa] [-assert] [-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] [-coveredfcov] [-cvg] [-pa]
[-assert] [-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. You can specify more than one coverage type with a single -code argument (example: “-code bcse”).
- **-codeAll**
(optional) Specifies that the command applies to all coverage types. Equivalent to -code bcstef or no specification of type at all.
- **-coveredfcov**
(optional) Save only non-zero covergroup bins and covered cover directives into the UCDB. Supported only 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. Supported only 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) Specifies that only test data to be saved into the UCDB. As a standalone function, you enter "coverage save -empty <my_ucdb_name>" at a Questa SIM TCL prompt, which produces a UCDB, with a default test record and a testname matching the specified name of the UCDB file. You can then load his file into viewcov mode and add other attributes to the test record or modify the default attributes, and then re-save the file.
- **-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 you can specify only one instance in Coverage View mode. You can also use <path> to specify a dataset other than the current dataset. (Refer to "[Object Name Syntax](#)" for instructions on how to specify a dataset.) If you do not specify a dataset, the current dataset is used. You can use only one dataset name per command invocation, or an error results.
- **-noadaptive**
 (optional) Eliminates storage of adaptive exclusion related information.
- **-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 you specify both -ovmtestname and -uvmtestname, the UVM testname is used.

For information on setting the testname within OVM/UVM, refer to “[Saving Coverage with UVM/OVM Test Name](#)” in the *Questa SIM User’s Manual*.

- **-pa**
Saves Power Aware-specific coverage data.
- **-prune <inst_path>**
(optional) Ignores hierarchies specified by <inst_path>. Can appear multiple times in a coverage save command.
- **-seed**
(optional) When used in conjunction with a simulation run with vsim -sv_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>). Supported only during live simulation, not in Coverage View mode. Not compatible with -testname.
For further details on setting the testname within UVM, refer to “[Saving Coverage with UVM/OVM Test Name](#)” in the User’s Manual.
- **<file_name>**
(required) Specifies the name of the database file to save. 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.

Description

The coverage save command is the preferred method of saving coverage data, but you can also save code coverage data with the \$coverage_save_mti (\$coverage_save) system task. (Refer to “SystemVerilog System Tasks and Functions” in the User’s Manual).

You can use this command and its arguments during live simulation or post-processing. Use “No Design” mode to save test data into a UCDB that you can subsequently load with the -empty switch.

The report displays code coverage data from generate blocks.

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 *somewhere.ucdb*) when the simulator exits:
coverage save -onexit somewhere
- 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
- Save Power Aware-specific coverage data.
coverage save -pa <name>.ucdb

Related Topics

[Code Coverage \[Questa SIM User's Manual\]](#)

[Coverage and Verification Management in the UCDB \[Questa SIM User's Manual\]](#)

[coverage analyze](#)

[coverage attribute](#)

[coverage report](#)

[coverage tag](#)

coverage tag

Creates (or deletes) links (tags) between objects in a coverage database to specific coverage objects in a testplan. A tag is a simple string associated with the object, and objects sharing the same tag name are linked together.

Syntax

Create links (tags)

```
coverage tag <UCDB_portion> -tagname <name> [<filters>]
```

Remove links

```
coverage tag <UCDB_portion> -delete -tagname <name> [-tagname <name>]... [<filters>]
```

Display links

```
coverage tag [<UCDB_portion>] [<filters>]
```

<UCDB_portion>

```
(-plansection <path> | -testrecord <test_name> | -path <path> [<path>...] | -duname <lib>.<duname>) [-recursive]
```

<filters>

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>]
```

```
<pattern_filters> = ([-match <string>] | [-cvgmatch <string>]) [-cvgininstmatch <string>] [-crossmatch <string>] [-crossmatch <string>]
```

```
<bin_filters> = -bins [-linerange [ln | ln-ln]...] [-item [bc]s] [<index> | <index>-<index>]...
```

Description

If you use the coverage tag command without the -tagname argument, it displays a list of tags, providing you with what is essentially a general purpose hierarchy browser.

You can also create links from the a coverage object back to the testplan through a process called “backlinking”.

Refer to “[Backlink Coverage Items to Plan with SV Attributes](#)” in the Verification Management User’s Manual for details.

This command is available only when you open a UCDB file with `vsim` -viewcov during post-simulation processing.

The coverage tag command defines the following:

- The portion of the UCDB to act upon. (Specify only one of the following for each use of the command):
 - the testplan (-plansection)
 - the context tree within the specified UCDB (-path)
 - the testrecords (-testrecord)
 - the design units (-duname)
- The action to be taken:
 - display tags
 - create tags (-tagname)
 - delete tags (-delete with -tagname)
- The objects to include 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 you use the -cvgmatch, -cvginstmatch, -cvpmatch, and/or -crossmatch switches 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 is: -cvpmatch, -crossmatch, -cvginstmatch, -cvgmatch. (Refer to the Examples section of this command.)

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 are selected when you use -bins. In order to select statement bins, you must also specify -code s.

Paths can contain wildcards, so take care not to select more than you intended. 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. You can specify more than one coverage type 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. Allows wildcards. 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. Allows wildcards. See “[coverage tag](#)” description for further details.
- **-cvpmatch <string>**
(optional) Specifies a match of a coverpoint against the specified covergroup instance or covergroup type. Allows wildcards. 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 used with -path) Applies the command to the specified library.design_unit(s). Allows wildcards. 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. Allows wildcards. 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 -cvgininstmatch.
- **-path <path> [<path>...]**
(required, unless specifying either -testrecord or -plansection, or used with -duname) Specifies a path to which the tag applies in the UCDB. You can specify more than one path. Allows wildcards. 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 “[xml2ucdb.ini Configuration File](#)” in the Verification Management User’s Manual. You can also use <path> 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. You can use only one dataset name per command invocation, or an error results.
- **-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. Allows wildcards.
- **-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 on, were specified for as many levels as are applicable. This switch is mutually exclusive with -match.

Tip

 Once a scope is tagged, none of its children or descendants will be, because coverage numbers always include descendants, so tagging descendants when writing a tag 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 specified tagname. Useful for determining all scopes (within the portion of the UCDB being acted upon) that share a particular tagname. Allows use of the “*” wildcard.
- **-tagname <name>**
(optional) Specifies the name of the tag to which the action is applied. Can be specified multiple times or not at all. Prints a list of all tags when <name> is not specified. If used for creation or deletion of tag(s) with -recursive, a coverage type filter is required (such as -cvg, -assert).
- **-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. Allows wildcards. Can be used only 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 there is 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. Accepts wildcard patterns, tagging any coverpoint-coverinstance-covergroup combination that matches:
coverage tag -tagname T1 -cvgmatch cvg1 -cvgininstmatch 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 \[Questa SIM User's Manual\]](#)

[Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)

[xml2ucdb.ini Configuration File \[Questa Verification Management User's Manual\]](#)

[Links Between the Plan Section and the Coverage Item \[Questa Verification Management User's Manual\]](#)

[coverage analyze](#)

[coverage attribute](#)

[coverage exclude](#)

[coverage ranktest](#)

[coverage goal](#)

[coverage report](#)

[coverage save](#)

[coverage unlinked](#)

coverage testnames

The coverage testnames command displays the testnames in the UCDB file currently loaded into memory. If a merged file is currently loaded, the command gives you a list of tests in the merged file.

Syntax

coverage testnames [[-tcl](#)]

Arguments

- [-tcl](#)
(optional) Print attribute information in a tcl format.

Description

This command is most useful when preparing to use the -testextract argument of coverage analyze or coverage report, because -testextract requires the test name. By default, the testname is the name of the UCDB file, though you can set it to whatever you like. Set the test name, before saving the UCDB file, with the command "coverage attribute -test mytestname".

This command is available only during post-simulation processing, when a UCDB file is opened with vsim -viewcov.

Related Topics

- [Code Coverage \[Questa SIM User's Manual\]](#)
- [Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)
- [coverage attribute](#)
- [coverage exclude](#)
- [coverage goal](#)
- [coverage ranktest](#)
- [coverage report](#)
- [coverage save](#)
- [coverage tag](#)
- [coverage analyze](#)
- [coverage weight](#)

coverage unlinked

Reports on objects that are “unlinked,” for test traceability purposes.

Tip

i When applying this command to design paths, it is a best practice 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> ]
<coverage type> = [ -assert ] [ -code { b | c | e | f | s | t }... ] [ -codeAll ] [ -cvg ] [ -directive ]
<filters> = [ -select instance ] [ -select name { -eq|-ne|-[n]regexp } { <string> } ] [ -select weight { -lt|-gt|-le|-ge|-eq|-ne } <int> ] [ -anyselected | -allselected ] [ -hideexcluded ] [ -nozeroweights ]
```

Description

The coverage unlinked command is used as a diagnostic tool to report on:

- testplan items 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.

This command is available only during post-simulation processing, when a UCDB file is opened with **vsim -viewcov**.

Arguments

- **-anyselected | -allselected**
(optional: Has an effect only with multiple -select options (does not work with -select instance).) These switches are mutually exclusive; only one may be given on each command line. Use any select criteria (logical OR) or all select criteria (logical AND) to determine what the report displays. Default is -allselected.
- **-append**
(optional) Appends to the output file, if -file option is used.
- **-assert**
(optional) Specifies that the command applies to assertion data.

- **-code {b | c | e | f | s | t}...**
(optional) Specifies that the command applies 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 that the command applies to all coverage types. Equivalent to -code bcestf.
- **-cvg**
(optional) Specifies that the command applies to covergroup data.
- **-directive**
(optional) Specifies that the command applies to directive data.
- **-du <duname>**
(optional) Restricts the report to design units matching the given specification, <duname>. Allows wildcards.
- **-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. You can combine this argument with -du to be du-relative. Use <path> to specify a dataset other than the current dataset. (See “[Object Name Syntax](#)” for instructions on how to specify a dataset.) If you do not specify a dataset, the current dataset is used. You can use only one dataset name per command invocation, or an error results.
- **-plansection <path>**
(optional) Restricts the command to the given testplan path. By default, the query checks 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 to restrict the query to the single object or objects specified in the command.

- **-select instance**
(optional) Includes only design instances 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 include 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 on, when compared with the specified weight <int>.
 - **-testname <test_name>**
(optional) Finds unlinked tests within a testplan. Allows wildcards. 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 \[Questa SIM User's Manual\]](#)
- [Verification Management Overview \[Questa Verification Management User's Manual\]](#)
- [Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)
- [coverage analyze](#)

[coverage attribute](#)

[coverage exclude](#)

[coverage goal](#)

[coverage ranktest](#)

[coverage report](#)

[coverage save](#)

[coverage tag](#)

[coverage testnames](#)

[coverage weight](#)

coverage weight

Sets or displays weights for global or individual items in the design.

Syntax

Setting global per-type weights

```
coverage weight [-bydu] [-bins] [-byinstance] [-type] [-attempted] [-active] [-disabled] [-fail] [-pass] [-vpass] [-code {b | c | e | f | s | t}... ] [-codeAll] [-assert] [-directive] [-feccond] <cond>]] [-fecexpr <expr>] [-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>.

Description

Use the coverage weight command to do any of the following:

- 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.

Enter a coverage weight command without an <integer_weight> to display weights.

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 or Verification Tracker Window. Refer to [Verification Management Browser](#) and [Verification Tracker Window](#) in the GUI Reference Manual for more information.

For more information regarding the type coverage statistics gathered, refer to [Calculation of Total Coverage](#) in the User's Manual.

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 no other assertion sub-types 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, in which case -bins is required to ensure that bins are matched. Otherwise, it matches functional coverage scopes only.
- **-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. You can specify more than one coverage type 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>.
- **-fecond <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 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>. You can use <path> to specify a dataset other than the current dataset. If you do not specify a dataset, the current dataset is used. You can use only one dataset name per command invocation, or an error results.
- **-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 \[Questa SIM User's Manual\]](#)

[Coverage and Verification Management in the UCDB \[Questa SIM User's Manual\]](#)

[Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)

[coverage analyze](#)

[coverage attribute](#)

[coverage exclude](#)

[coverage goal](#)

[coverage ranktest](#)

[coverage report](#)

[coverage save](#)

[coverage tag](#)

[coverage testnames](#)

[coverage unlinked](#)

dataset alias

Maps an alternate name (alias) to an open dataset.

Syntax

dataset alias <dataset_name> [<alias_name>]

Description

A dataset can have any number of unique aliases, but all dataset names and aliases must be still be unique when more than one dataset is open. Aliases are not saved to the .wlf file, and you must remap them if you close and re-open the dataset.

Arguments

- <dataset_name>
(required) Specifies a dataset name or currently assigned dataset alias. Must be 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. Allows wildcard characters.

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

Removes all event data from the current simulation WLF file, while retaining all currently logged signals. Subsequent run commands continue to accumulate data in the WLF file.

Note

 This command applies only to WLF-based simulation datasets.

Syntax

dataset clear

Description

This command has no effect on coverage (UCDB) datasets.

Running this command with no design loaded returns the error “Dataset not found:sim”. If you run the command with a design loaded, the “sim:” dataset is cleared, regardless of which dataset is currently set. Clearing the dataset clears any open Wave window based on the “sim:” dataset.

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 \[Questa SIM User's Manual\]](#)

dataset close

Closes an active dataset.

Note

 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 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

Configures WLF parameters for an open dataset and all aliases mapped to that dataset.

Note

 This command has no effect on coverage datasets (UCDB).

Syntax

```
dataset config <dataset_name> [-wlfcachesize [<n>]] [-wlfdeleteonquit [0 | 1]] [-wlfopt [0 | 1]]
```

Arguments

- <dataset_name>
(required) Specifies an open dataset or dataset alias to configure. Must be 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 info](#)

[dataset list](#)

[dataset open](#)

[dataset rename](#)

[dataset restart](#)

[dataset save](#)

[dataset snapshot](#)

[WLF File Parameter Overview \[Questa SIM User's Manual\]](#)

dataset current

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 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 \[Questa SIM User's Manual\]](#)

dataset info

Reports a variety of information about a dataset.

Syntax

dataset info {name | file | exists} <dataset_name>

Arguments

- {name | file | exists}

(required) Identifies what type of information to report.

Allows only one option per command. 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 is not.

Must be 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

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

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 you specify.

Syntax

```
dataset open <file_name> [<dataset_name>]
```

Description

The file can be the existing WLF file for a currently running simulation. To close a dataset, use the [dataset close](#) command.

Arguments

- <file_name>
(required) Specifies the file to open as a view-mode dataset. Must be 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 name identifies the dataset in the current session. By default the dataset prefix is 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](#)

dataset rename

Changes the name of a dataset to the new name you specify.

Syntax

`dataset rename <dataset_name> <new_dataset_name>`

Arguments

- `<dataset_name>`
Specifies the existing name of the dataset. Must be the first argument.
- `<new_dataset_name>`
Specifies the new name for the dataset. Must be the second argument.

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

Unloads the specified dataset or currently active dataset and reloads the dataset using the same dataset name.

Note

 The contents of the 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 restarts.

Related Topics

[dataset alias](#)
[dataset clear](#)
[dataset close](#)
[dataset config](#)
[dataset info](#)
[dataset list](#)
[dataset open](#)
[dataset rename](#)
[dataset save](#)
[dataset snapshot](#)

dataset save

Writes data from the current simulation to a specified file while the simulation is in progress.

Note

 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. Must be the first argument.
- `<file_name>`
(required) Specifies the name of the file to save. Must be the second argument.

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

Saves data from the current WLF file (*vsim.wlf* by default) at a specified interval. Provides you with sequential or cumulative "snapshots" of your simulation data.

Note

 This command does not apply to coverage datasets (UCDB).

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 to save the files. Either absolute or relative paths can be used. Default is to save to the current working directory.
- **-disable**
(optional) Turns snapshotting off. Stores all dataset snapshot settings from the current simulation 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; the default is *vsim_snapshot.wlf*. The suffix *.wlf* is appended to the specified filename. Some arguments may also append 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 — Creates a new file for each snapshot. Adds an incrementing suffix (1 to n) 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 — Clears the current WLF file every time a snapshot is taken.

- -report
 - (optional) Lists current snapshot settings in the Transcript window and ignores all other options.
- -reset
 - (optional) Resets values back to defaults, then applies the remainder of the arguments on the command line. See Examples section, below. If specified 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 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 the final argument to the dataset snapshot command.
 <n> — Any positive integer.
 <unit> — (optional) A suffix specifying a unit of time. The default is to specify the current simulation resolution by omitting <unit>. If you specify <unit>, you must enclose <limit> and <unit> within braces ({}). Valid time units are: fs, ps, ns, us, ms, sec, min, and hr.

Examples

- Create the file *vsim_snapshot_<n>.wlf*, and write to it every time the current WLF file reaches a multiple of 10 MB (that is, at 10 MB, 20 MB, 30 MB, and so on.).
dataset snapshot -size 10
- Similar to the previous example, but in this case, clear the current WLF file 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

Removes objects from either the List or Wave window.

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.) Uses the default window, if none is specified. 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. Specify multiple object names as a space-separated list. Allows wildcard characters. Must be 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*
```

describe

Displays information about simulation objects and design regions in the Transcript window.

Syntax

```
describe <name>...
```

Description

This command displays information about the following types of simulation objects and design regions:

- **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 both of the following are true:

- You are running C Debug. ([C Debug](#) is described in more detail in the User's Manual.)
- 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](#)” in the User's Manual.

Arguments

- <name>...

(required) The name of an HDL object, SystemC signal, or C variable for which you want a description.

Specify multiple object names as a space separated list. Allows wildcard characters. 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
```

disablebp

Turns off breakpoints and when commands.

Note

 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

[enablebp](#)

[onbreak](#)

disable_menu

Disables the specified menu within the specified window, making it inactive (grayed-out).

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. Must be the first argument.
- <menu_path>
(required) Name of the Tk menu-widget path. Must be the final argument.

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

[disable_menuitem](#)

[enable_menu](#)

disable_menuitem

Disables a specified menu item within the specified menu path of the specified window. The menu item becomes inactive (grayed-out).

Syntax

`disable_menuitem <window_name> <menu_path> <label>`

Arguments

- `<window_name>`

(required) Tk path of the window containing the menu. Must be the first argument to the command.

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 can include a submenu as shown in the example below. Must be the second argument.

- `<label>`

(required) Menu item text. Must be the final argument.

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

[disable_menu](#)

[enable_menu](#)

do

Executes the commands contained in a DO file.

Syntax

`do <filename> [<parameter_value>...]`

Description

A DO file can have any name and extension. Any encountered errors interrupt the DO file script execution, unless you specify a `onerror` command, or the `OnErrorHandler` Tcl variable with the `resume` command. You can use the `onbreak` command to take action with source code breakpoint cases.

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 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. You must separate multiple parameter values 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](#)” in the User’s Manual.

Note



There is no limit on the number of parameters that can be passed to macros, but 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 \[Questa SIM User's Manual\]](#)

[General Modes of Operation \[Questa SIM User's Manual\]](#)

[Using a Startup File \[Questa SIM User's Manual\]](#)

[DOPATH \[Questa SIM User's Manual\]](#)

[Saving a Transcript File as a DO File \[Questa SIM GUI Reference Manual\]](#)

down

Searches for object transitions or values in the specified List window.

Syntax

```
down [-expr <expression>] [-falling] [<n>] [-noglitch] [-rising] [-value <sig_value>] [-window <wname>]
```

Description

The down command searches on objects that are 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, to find the time when an object takes on a particular value, or the time when an expression of multiple objects evaluates to true. See the [up](#) command for related functionality.

There are three steps to using down:

1. Click the desired object.
2. Click 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>

Arguments

- **-expr <expression>**

(optional) Searches the List window until the expression evaluates to a boolean true condition.

<expression> — An expression that involves one or more objects, but is limited to objects that have been logged in the referenced List window. You can specify an object 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 enclosed in braces.

- **-falling**

(optional) Searches for a falling edge on the specified object, if that object is a scalar object. Ignores non-scalar objects.

- **<n>**

(optional) Specifies to find the nth match. If less than n are found, returns the number found and a warning message, and the marker is positioned at the last match.

- **-noglitch**
(optional) Specifies to ignore delta-width glitches.
- **-rising**
(optional) Searches for a rising edge on the specified object, if that object is a scalar object. Ignores non-scalar objects.
- **-value <sig_value>**
(optional) Specifies the value of the object to match.
 <sig_value> — A value specified in the same radix as the selected object. Case is ignored, but otherwise the value must be an exact string match. Don't-care bits are not supported.
- **-window <wname>**
(optional) Specifies a non-default instance of the List window. Uses the default List window if <wname> is not specified. 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)}

drivers

Displays the names and strength of all drivers of the specified object.

Syntax

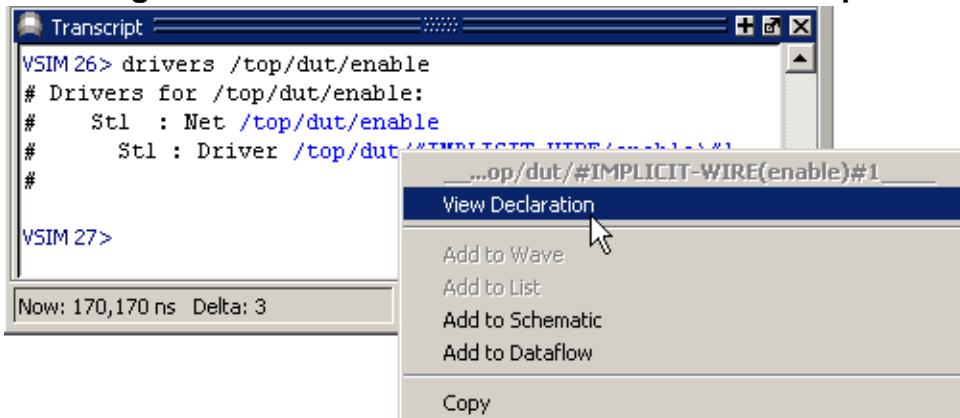
```
drivers <object_name> [-source]
```

Description

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 appears 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 2-3. drivers Command Results in Transcript



Arguments

- <object_name>
(required) Specifies the name of the signal or net for which to display drivers. All signal or net types are valid. Accepts multiple names and wildcards.
- -source
(optional) Returns the source file name and line number for each driver of the specified signal or net. Returns the value n/a when unable to determine the source location for the 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.

dumplog64

Dumps the contents of the specified WLF file to stdout in a readable format.

Note

 When you use this command, you cannot open the WLF file for writing in a simulation. You cannot use this command in a DO file.

Syntax

`dumplog64 <filename>`

Arguments

- `<filename>`
(required) The name of the WLF file to read.

echo

Displays a specified message in the Transcript window.

Syntax

```
echo [<text_string>]
```

Arguments

- <text_string>

(required) Specifies the message text to display. If you surround the text string with quotation marks, blank spaces display as entered. If you omit quotation marks, adjacent blank spaces are compressed into a single 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 adjacent blank spaces 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

Invokes the editor specified by the EDITOR environment variable. By default, the specified filename opens in the Source window.

Syntax

`edit [<filename>]`

Arguments

- `<filename>`

(optional) Specifies the name of the file to edit. If you omit the `<filename>` argument, the editor opens the current source file. If you specify a non-existent filename, the editor opens a new file. You can use either absolute or relative paths.

Related Topics

[notepad](#)

[EDITOR \[Questa SIM User's Manual\]](#)

enablebp

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

[disablebp](#)

[onbreak](#)

enable_menu

Enables a previously disabled menu, changing it from grayed-out to normal and responsive.

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

[disable_menu](#)

enable_menuitem

Enables a previously disabled menu item, changing it from grayed-out to normal and responsive.

Syntax

`enable_menuitem <window_name> <menu_path> <label>`

Arguments

- `<window_name>`
(required) Tk path of the window containing the menu. Must be the first argument.
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 can include a submenu, as shown in the example below. Must be the second argument.
- `<label>`
(required) Menu item text. Must be the final argument.

Examples

Locate the mywindow window and enable 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

[disable_menuitem](#)

encoding

Translates between the 16-bit Unicode characters used in Tcl strings and a named encoding, such as Shift-JIS. Consists of several related commands.

Syntax

```
encoding convertfrom <encoding_name> <string>
encoding convertto <encoding_name> <string>
encoding names
encoding system <encoding_name>
```

Description

The encoding commands work with the encoding of your character representations in the GUI:

- `encoding convertfrom` — Converts a string from the named encoding to Unicode.
- `encoding convertto` — Converts a string to the named encoding from Unicode.
- `encoding names` — Returns a list of all valid encoding names (takes no arguments).
- `encoding system` — Changes the current system encoding to a named encoding. If you omit a new encoding the command returns the current system encoding. The system encoding is used whenever Tcl passes strings to system calls.

Arguments

- `string`
Specifies a string to be converted.
- `encoding_name`
The name of the encoding to use.

environment

Allows you to display or change the current dataset and region/signal environment.

Note

 You can abbreviate this command as “env”, as shown in the Examples section, below.

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 displays the pathname of the current region/signal environment.
- `-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

[Setting your Context by Navigating Source Files \[Questa SIM GUI Reference Manual\]](#)

examine

Examines one or more objects and displays current values (or the values at a specified previous time) in the Transcript window.

Note

 You can abbreviate the examine command as “exa”.

Syntax

```
examine <name>... [-delta <delta>] [-env <path>] [-event <time>] [-handle] {[-in] [-out] [-inout] | [-ports]} [-internal] [-expr <expression>] [-name] [-<radix_type>] [-radix <radix_type>][,<radix_flag>][,...]] [-radixenumnumeric | -radixenumsymbolic] [-showbase | -noshowbase] [-time <time>] [-value]
```

Description

The examine command can also compute the value of an expression of one or more objects.

If you are optimizing the design with [vopt](#), some of the objects listed below may not be available for viewing. Refer to "[Preservation of Object Visibility for Debugging](#)" in the User's Manual for more information.

If you are using C Debug, examine can display the value of a C variable as well.

You can examine the following objects:

- **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. Refer to "[SystemC Object and Type Display](#)" in the User's Manual for specific information related to viewing SystemC objects.

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 can also be examined within the GUI (actual signals are examined in the kernel).

The examine command uses the following rules 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 are 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, with the result that some objects that may be visible from a given context will not be found when using wildcards 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 never matches a path separator. For example, */dut/** matches */dut/siga* and */dut/clk*, while */dut** does not match either of those.

If you do not specify a radix 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. Refer to [DefaultRadix](#) in the User's Manual.
- Set the default radix flags value with the radix command or by editing the DefaultRadixFlags variable in the modelsim.ini file. Refer to [DefaultRadixFlags](#) in the User's Manual.
- 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.

Arguments

- <name>...

(required except when specifying -expr.) Specifies the name of any HDL or SystemC object. Allows all object types except those of the type file. Accepts multiple names and wildcards. 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. The default is 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 is 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 does 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.

- -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 returns 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, signed, sfixed, symbolic, time, ufixed, unsigned, and default.

This option overrides the global setting of the default radix. (Refer to the description of the modelsim.ini [DefaultRadix](#) variable in the User's Manual).

- **-radix [<radix_type>][,<radix_flag>][,...]**

(optional) Specifies the radix and/or the radix flags to be used by the examine command. You can use the **-radix <radix_type>** switch 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, pretty, 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. Specify multiple flags as a comma separated list. Must follow **-<radix_type>** when the two are specified together.

Valid radix flags:

Table 2-5. Radix flag Arguments to the Examine Command

Argument	Description
enumnumeric	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: <ul style="list-style-type: none"> • d = decimal • b = binary • h = hexadecimal • a = ASCII • t = time

For example, instead of simply displaying a vector value of “31”, a value of “16'h31” can 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 (Refer to the descriptions of the modelsim.ini [DefaultRadix](#) and [DefaultRadixFlags](#) in the User's Manual).

- **-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 | -noshowbase**
(optional) Enables or disables displaying the number of bits of the vector and the radix used (binary = b, decimal = d, hexadecimal = h, ASCII = a, and time = t).
For example, with **-showbase**, 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 enclosed 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 is 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 is the following one-element list:

```
# {"NOT ELABORATED"}
```

- If the file is closed, the result is the following one-element list.

```
# {"CLOSED"}
```

- In all other cases, the result is 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 in 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 (that is, 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 between the closing '\' and 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. Quotation marks (" ") are required when spaces are involved; using quotation marks instead of braces causes the Tcl interpreter to expand variables before calling the command.

examine -time \$t -name \${fifo}{1 to 3}" \${fifo}{1}
- Because **-time** is not specified, this expression is 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 (a blank-separated list of strings). You can enter commands such as:

```
foreach sig [find sig ABC*] {echo "Signal $sig is [exa $sig]" ...}  
if {[examine -bin signal_12] == "11101111XXXZ"} {...}  
examine -hex [find *]
```

You can also use the Tcl variable array **\$examine()** 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. (Refer to [Source Window](#) in the GUI Reference Manual for more information.)

- 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

[DefaultRadix](#) [Questa SIM User's Manual]

exit

Exits the simulator and the Questa SIM application.

Syntax

`exit [-force] [-code <integer>]`

Description

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.

Arguments

- `-force`
(optional) Quits without asking for confirmation. If you omit this argument, 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 "[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. You can use the [onbreak](#) command to specify commands to execute 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

Enables, disables, and sets coverage targets for SystemVerilog and PSL cover directives.

Syntax

```
fcover configure <coverage_directive>... [-at_least <count>] [-disable] [-enable] [-exclude]  
[-include] [-limit <count>] [-log on | off] [-recursive] [-weight <integer>]
```

Description

Invoke this command without any optional arguments to report current settings for the specified directive(s).

Arguments

- <coverage_directive>...
(required) Identifies the functional coverage directive(s) to which to apply the configuration parameters. You can specify multiple directives. Allows wildcards.
- -at_least <count>
(optional) Specifies a target count for the selected coverage point(s).

 <count> — Any positive integer. 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. Refer to [CoverAtLeast](#) in the User's Manual for more information.
- -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. Refer to [CoverEnable](#) in the User's Manual for more information.
- -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 still show up in the Cover Directives window, but they do not count toward coverage totals, and do not 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**. The default is -1 (unlimited).

The value for **-limit** automatically adjusts 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. Refer to [CoverLimit](#) in the User's Manual for more information.

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. Refer to [CoverLog](#) in the User's Manual for more information.

- **-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 has twice the impact on the aggregated coverage numbers as a directive with a weight of 1. If its weight is set to 0 but a directive is otherwise enabled, its coverage does 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 \[Questa SIM User's Manual\]](#)

[CoverAtLeast \[Questa SIM User's Manual\]](#)

[CoverEnable \[Questa SIM User's Manual\]](#)

[CoverLimit \[Questa SIM User's Manual\]](#)

[CoverLog \[Questa SIM User's Manual\]](#)

[Verification with Functional Coverage \[Questa SIM User's Manual\]](#)

[Verification with Assertions and Cover Directives \[Questa SIM User's Manual\]](#)

find

Locates objects by type and name.

Note



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] [-protected] [-recursive] [-nodu]
find virtuals <object_name> ... [-kind <kind>] [-unsaved] [-recursive]
find classes [<class_name>]
find objects [-class <class_name>] [-isa <class_name>] [<object_name>]
```

Description

The find command uses the following rules to locate an object:

- If the name does not include a dataset name, use the current dataset.
- If the name does not start with a path separator, use the current context.
- If the name is a path separator followed by a name that is not the name of a top-level design unit, use the first top-level design unit in the design.
- For a relative name containing a hierarchical path, if the first object name cannot be found in the current context, then do an upward search 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 do an upward search 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, if you use wildcards, objects that are visible from a given context, but are within a higher enclosing scope, may not be found.
- 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 never matches a path separator. For example, /dut/* matches /dut/siga and /dut/clk, but /dut* does not.
- Because square brackets are wildcards in the find command, only parentheses (()) can be used to index or slice arrays.

- The find command uses the *WildcardFilter* Tcl preference variable to exclude the specified types of objects when performing the search.

See [Design Object Names](#) for more information on specifying names.

Arguments

Arguments to the command are grouped by object type.

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.

Note

 The find nets and find signals commands are also supported in vsim -batch mode.

- <object_name> ...
(required) Specifies the net or signal for which you want to search. Allows multiple nets and signals and wildcard characters. 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 includes ports of mode IN.
- -inout
(optional) Specifies that the scope of the search includes ports of mode INOUT.
- -internal
(optional) Specifies that the scope of the search includes internal (non-port) objects.
- -nofilter
(optional) Specifies to ignore the *WildcardFilter* Tcl preference variable when finding signals or nets.
- -out
(optional) Specifies that the scope of the search includes ports of mode OUT.
- -ports
(optional) Specifies that the scope of the search includes 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.

Note

 The find instances command is also supported in vsim -batch mode.

- **-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 to search for. Allows multiple instances and wildcard characters. Mutually exclusive with **-file** and **-bydu**.

- **-protected**

Filters and reduces the output to only those instances (implied by the **<object_name>** argument) that are protected because they meet one or both of the following requirements:

- The instance is in a protected region.
- The instance is in a design unit that has a protected region in it.

- **-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 design unit name from any instance in the output.

Arguments for virtuals

When searching for virtuals, you must specify all optional arguments before any object names.

- **<object_name> ...**

(required) Specifies the virtual object to search for. Allows multiple virtuals and wildcard characters.

- **-kind <kind>**

(optional) Specifies the kind of virtual object to search for.

<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 to search. Allows wildcard characters . 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 to search for. Allows wildcard characters. 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.

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/c1*

- 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, you must include 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 is typically 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 in a specific display window, use the [search](#) command or select **Edit > Find**.

find connections

Returns the set of nets that are electrically equivalent to a specified net. Available only during a live simulation.

Syntax

find connections <net>

Arguments

- <net>
(required) A net in the design.

Examples

find connections /top/p/strb

returns:

```
# Connected nets for strb
#   output : /top/p/strb
#   internal : /top/pstrb
#   input : /top/c/pstrb
```

find drivers

Prerequisite:

Before using this command, you need to do either of the following:

- Run the vopt command with the -debugdb argument.
- If **vopt** is not used, simulate the design with the -debugdb and -voptargs="+acc" arguments to the **vsim** command.

Traces backward in time to find the active driver(s), processes, or first elements of the specified signal or signal event.

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 -chosex [-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>
```

Description

The **find drivers** 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. (Refer to [Initiating Causality Traceback from the GUI](#) in the User's Manual.)
- 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 can 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 Examples section for more information.

You can use the Escape key to interrupt the various find drivers operations.

You can enter `find drivers -help` or `find drivers -?` to access a help menu.

Note

 Full causality traceback functionality requires optimization of your design with `vopt` or `vsim -voptargs`. Refer to [Using Causality Traceback](#) in the User's Manual for more information.

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. Allows special characters, except a backslash (\).
- **-schematic**
(optional) Places all signals contained in the path found by the trace 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) Displays all of the possible driving assignments for a signal (does not use simulation results). 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, scrolls to show that line, and highlights the driving signal. If the trace type is -possible, 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. The default is to specify the current simulation resolution by omitting <unit>. You must enclose <time> and <unit> within braces ({}). Valid VHDL time units are: fs, ps, ns, us, ms, sec, min, and hr.
- **-transcript**
(optional) Reports trace results 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.

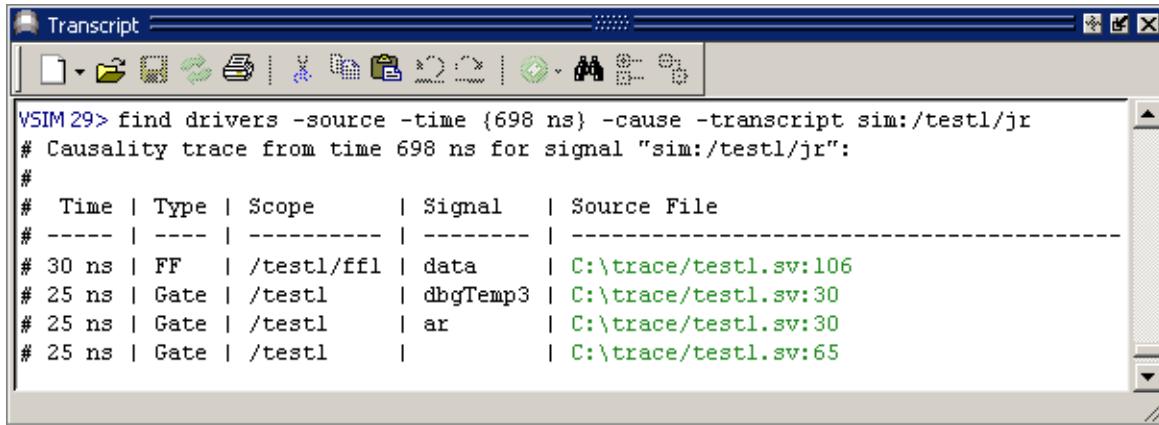
- **-wave**
(optional) Adds all signals in the path found by the trace to a dedicated Wave window, and displays cursors to show the beginning and ending times of the trace. Clears the dedicated Wave window of signals before displaying the results of a new trace.
- **-width <n>**
(optional) Specifies 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 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
```



The screenshot shows a Windows-style Transcript window titled "Transcript". The window contains the command "VSIM 29> find drivers -source -time {698 ns} -cause -transcript sim:/test1/jr" followed by its output. The output is a table listing causalities for the signal "sim:/test1/jr" at time 698 ns. The table has columns for Time, Type, Scope, Signal, and Source File. The data is as follows:

Time	Type	Scope	Signal	Source File
30 ns	FF	/test1/ff1	data	C:\trace/test1.sv:106
25 ns	Gate	/test1	dbgTemp3	C:\trace/test1.sv:30
25 ns	Gate	/test1	ar	C:\trace/test1.sv:30
25 ns	Gate	/test1		C:\trace/test1.sv:65

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 \[Questa SIM User's Manual\]](#)

find infiles

Sets up a search for a string in the specified file(s) and prints the results to the Transcript window. The results have individual hotlinks that will open the file and display the location of the string.

Syntax

`find infiles <string_pattern> <file>...`

Description

When you run this command in command-line mode from outside of the GUI, the results are sent to stdout with no hotlinks.

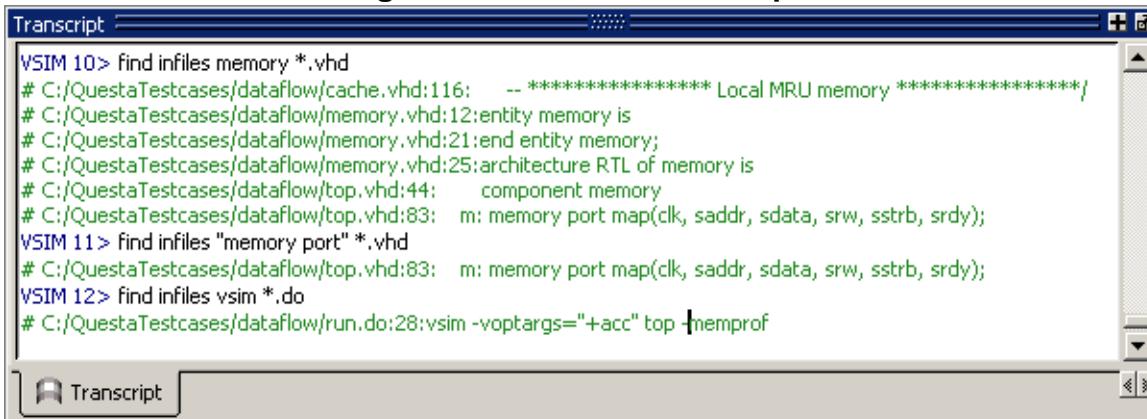
Arguments

- `<string_pattern>`
(required) The string to search for. Must be the first argument. You can use Tcl regular expression wildcards to further restrict the search capability.
- `<file>...`
(required) The file(s) to search. Must be the second argument. You can use Tcl regular expression wildcards to further restrict the search capability.

Examples

Figure 2-4 shows a screen capture containing examples and results of the find infiles command.

Figure 2-4. find infiles Example



The screenshot shows a Windows-style application window titled "Transcript". The window contains a scrollable text area displaying command-line output. The output is as follows:

```
VSIM 10> find infiles memory *.vhd
# C:/QuestaTestcases/dataflow/cache.vhd:116: -- **** Local MRU memory ****
# C:/QuestaTestcases/dataflow/memory.vhd:12:end entity memory;
# C:/QuestaTestcases/dataflow/memory.vhd:21:entity memory is
# C:/QuestaTestcases/dataflow/memory.vhd:25:architecture RTL of memory is
# C:/QuestaTestcases/dataflow/top.vhd:44: component memory
# C:/QuestaTestcases/dataflow/top.vhd:83: m: memory port map(clk, saddr, sdata, srw, sstrb, srdy);
VSIM 11> find infiles "memory port" *.vhd
# C:/QuestaTestcases/dataflow/top.vhd:83: m: memory port map(clk, saddr, sdata, srw, sstrb, srdy);
VSIM 12> find infiles vsim *.do
# C:/QuestaTestcases/dataflow/run.do:28:vsim -voptargs="+acc" top +memprof
```

find insource

Searches for a string in the source files for the current design and prints the results to the Transcript window, with individual hotlinks to open the source and display the location of the string.

Note

 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 quotation marks (" ") if it includes spaces. You must specify the <pattern> at the end of the command line; any switches specified after <pattern> are not registered.

- -exact | -glob | -regexp

(optional) Defines the style of regular expression used in the <pattern>

-exact — Indicates that no characters have special meaning, 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 hotspot feature, but can simplify post-processing.

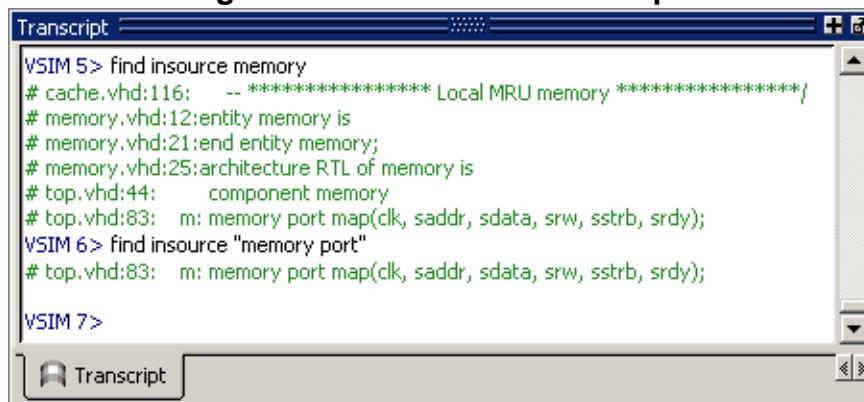
- -nocase

(optional) Treats <pattern> as case-insensitive.

Examples

- [Figure 2-5](#) shows examples of the find insource command, including results.

Figure 2-5. find insource Example



The screenshot shows the Questa SIM Transcript window. The transcript area contains the following text:

```
VSIM 5> find insource memory
# cache.vhd:116:  .. **** Local MRU memory ****/
# memory.vhd:12:entity memory is
# memory.vhd:21:end entity memory;
# memory.vhd:25:architecture RTL of memory is
# top.vhd:44:  component memory
# top.vhd:83:  m: memory port map(clk, saddr, sdata, srw, sstrb, srdy);
VSIM 6> find insource "memory port"
# top.vhd:83:  m: memory port map(clk, saddr, sdata, srw, sstrb, srdy);

VSIM 7>
```

- 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

[DISABLE_ELAB_DEBUG \[Questa SIM User's Manual\]](#)

force

Enables you to apply stimulus interactively to VHDL signals, Verilog nets and registers, and SystemC boundary types.

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

Specifying this command without arguments 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, if you enter:

```
force -freeze /top/p/addr 0 100, 1 150 -r 200 -cancel 2000
```

with the specified times being relative to the current simulation time of 2820 ns,

Entering:

```
force
```

Returns:

```
# force -freeze /top/p/addr 0 {@2920 ns} , 1 {@2970 ns}  
      -repeat {@3020 ns} -cancel {@4820 ns}
```

Note When you run the force command, the simulator translates the relative time you specify into absolute time.

Description

You can create a complex sequence of stimuli when using the force command in a DO file.

The constraints on what you can and cannot force include:

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](#)” in the User’s Manual for more information.
- Virtual Signals if the number of bits corresponds to the signal value. Refer to “[Virtual Signals](#)” in the User’s Manual.
- 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.

You can use the -help switch to find additional information on this command.

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 can 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 the first argument to the **force** command.
- <value>

(required when forcing a value change) Specifies the value to force the object to. The specified value must be appropriate for the type. Must be 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

To force a negative value, ensure that the negative sign ‘-’ precedes the base designation. For example, to force a -2 value in base 10 in VHDL, write:

```
force <object_name> -10#2
```

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 table in the *pref.tcl* file controls the translation. 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. When used for registers, it behaves like the [change](#) command. Supports multi-dimensional indexing on the force target.

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. (Refer to [DefaultForceKind](#) in the User’s Manual.)

- -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. (Refer to [DefaultForceKind](#) in the User's Manual.)

- -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}
-repeating {@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

[DefaultForceKind \[Questa SIM User's Manual\]](#)

[Force and Release Statements in Verilog \[Questa SIM User's Manual\]](#)

[Force Command Defaults \[Questa SIM User's Manual\]](#)

[noforce](#)

[Virtual Signals \[Questa SIM User's Manual\]](#)

formatTime

Provides global format control for all time values displayed in the GUI. When specified without arguments, 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 displays 6,458,131 ps.

- Use largest unit value possible.

formatTime +bestunits

Displays 8 us instead of 8,000 ns.

fsm list

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 \[Questa SIM User's Manual\]](#)

fsm properties

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](#)

[Finite State Machines \[Questa SIM User's Manual\]](#)

fsm view

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](#)

[Finite State Machines \[Questa SIM User's Manual\]](#)

gc configure

Prerequisite:

Before using this command, do one of the following:

- Regular simulation — Simulate with the vsim command, but omit the -classdebug argument.
- Interactive class debugging — Simulate with the vsim -classdebug command.

Specifies when to run the System Verilog garbage collector. 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 or disables post-simulation run garbage collection.
 - 0 — Off, default for regular simulation
 - 1 — On, default for interactive class debugging
- **-onstep 0 | 1**
(optional) Enables or disables post-step garbage collection.
 - 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 that, when exceeded, causes the garbage collector to run.
 - <n> — Any positive integer where <n> is the number of megabytes.
 - Regular simulation default = 100 megabytes
 - Interactive class debugging default = 5 megabytes

Description

You can configure the garbage collector 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).

Related Topics

[SystemVerilog Class Debugging \[Questa SIM User's Manual\]](#)

[Class Instance Garbage Collection \[Questa SIM User's Manual\]](#)

[GCThreshold \[Questa SIM User's Manual\]](#)

[GCThresholdClassDebug \[Questa SIM User's Manual\]](#)

gc run

Runs the SystemVerilog garbage collector.

Syntax

gc run

Arguments

None

Related Topics

[gc configure](#)

[SystemVerilog Class Debugging \[Questa SIM User's Manual\]](#)

[Class Instance Garbage Collection \[Questa SIM User's Manual\]](#)

[GCThreshold \[Questa SIM User's Manual\]](#)

[GCThresholdClassDebug \[Questa SIM User's Manual\]](#)

gdb dir

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 you do not specify a directory, the source directory search path is set to the gdb default—\$cdir:\$cwd. You can use either absolute or relative paths. Specify multiple paths as a space-separated list. Allows wildcards and relative paths.

Examples

Set the source directory search paths to `./dut/` and `../foo/`.

`gdb dir ./dut/ ../foo/`

Related Topics

[C Debug \[Questa SIM User's Manual\]](#)

[Setting Up C Debug \[Questa SIM User's Manual\]](#)

getactivecursortime

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](#)

getactivemarkertime

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](#)

help

Displays a brief description and syntax for the specified command in the Transcript window.

Syntax

help [[<command>](#) | [<topic>](#)]

Arguments

- <command>
(optional) Specifies the command to provide help for. The entry is case and space sensitive.
- <topic>
(optional) Specifies a topic to provide help for. 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

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.

jobs

Controls JobSpy, a tool for monitoring and controlling batch simulations and simulation farms.

Syntax

`jobs` [`-gui` | `-startd` | `-killd` | `jobs` | `status` | `<jobid>` `<command>`]

Note

 Entering `jobs` on the command line with no arguments provides additional usage information.

Arguments

- `-gui`
(optional) Launches the JobSpy Job Manager GUI.
- `-startd`
(optional) Starts the `jobs` daemon which enables job tracking. You must 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](#)” in the User’s Manual for further details.
- `-killd`
(optional) Terminates the JobSpy daemon.
- `jobs`
(optional) Returns a list of current jobs with a variety of status information (for example, job ID, current simulation time, start time, and so on).
- `status`
(optional) Returns the status of the JobSpy daemon.
- `<jobid>` `<command>`
(optional) Specifies a job ID to be processed by `<command>`. Use `jobs` to get a list of job IDs.
`<command>` — A JobSpy simulator command. Refer to “[Simulation Commands Available to JobSpy](#)” in the User’s Manual for a list of valid commands.

Related Topics

[Monitoring Simulations with JobSpy \[Questa SIM User’s Manual\]](#)

[Running the JobSpy GUI \[Questa SIM User’s Manual\]](#)

[Simulation Commands Available to JobSpy \[Questa SIM User’s Manual\]](#)

[Start the JobSpy Daemon \[Questa SIM User's Manual\]](#)

layout

Enables you to perform a number of editing operations on custom GUI layouts, such as loading, saving, maximizing, and deleting.

Syntax

```
layout active
layout current
layout delete <name>
layout load <name>
layout names
layout normal
layout maximized
layout restoretpe <window>
layout save <name>
layout showsuppresstypes
layout suppresstype <window>
layout togglezoom
layout zoomactive
layout zoomwindow <window>
```

Description

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^{®1}) 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

1. Linux[®] is a registered trademark of Linus Torvalds in the U.S. and other countries.

-
- layout restoretpe — 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

Arguments

- <name>
(required) Specifies the name of the layout.
- <window>
(required) The window specification can be any format accepted by the [view](#) command. You can specify the window by its type (such as wave, list, objects), by the windowobj name (such as main_pane.wave, .main_pain.library), or by the tab name (such as wave1, list3)

Related Topics

[Simulator GUI Layout Customization \[Questa SIM GUI Reference Manual\]](#)

[Configuring Default Windows for Restored Layouts \[Questa SIM GUI Reference Manual\]](#)

lecho

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. Specify multiple entries as a space-separated list.

Examples

- Print the Wave window configuration list to the Transcript window.

lecho [configure wave]

left

Searches left (previous) for signal transitions or values in the specified Wave window.

Syntax

```
left [-expr {<expression>}] [-falling] [<n>] [-rising] [-value <sig_value> [-noglitch]]  
[-window <wname>]
```

Description

This command executes a 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 the desired waveform.
2. Click the desired starting location. (You can also use the [seetime](#) command to initially position the cursor from the command line.)
3. Issue the left command.

Returns: <number_found> <new_time> <new_delta>

Arguments

- **-expr {<expression>}**

(optional) Searches the waveform display until the expression evaluates to a boolean true condition.

<expression> — An expression that involves one or more objects, but is limited to objects that have been logged in the referenced waveform display. An object can 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. You must enclose the expression in 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, this option is ignored.

- **<n>**

(optional) Specifies to find the nth match. The default is 1. If less than n are found, returns the number found with a warning message, and positions the cursor 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, this option is ignored.
- **-value <sig_value>**
(optional) Specifies the value of the object to match.

`<sig_value>` — A value specified in the same radix as the selected object. Case is ignored, but otherwise the value must be an exact string match. Does not support don't-care bits. 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. Uses the default Wave window when `<wname>` is not specified. 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

 You can also use the “[Wave Window Mouse and Keyboard Shortcuts](#)” described in the GUI Reference Manual for next and previous edge searches. Tab searches right (next) and shift-tab searches left (previous).

log

Creates a wave log format (WLF) file containing simulation data for all HDL objects whose names match the provided specifications.

Syntax

```
log [-class <classtype>] [-howmany] [-filter <f> | -nofilter <f>] {[[-in] [-inout] [-out] | [-ports]]  
[-internal] [-mvcall] [-mvcovm] [-mvcrecccomplete] [-optcells] [-recursive] [-depth <level>]}  
<object_name> ...  
log -flush [<object>]
```

Description

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 automatically records 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 a Questa Verification IP, as well as transaction instances, are not logged by default: you must log them explicitly through the [log](#), [add wave](#), or [add list](#) commands. You can log completed transaction instances from Questa Verification IP using -mvcrecccomplete (all except add list), and transaction streams using -mvcall or -mvcovm.

Note

 The log command is also known as the "add log" command.

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

 Use of 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 log command can take as many [-filter <f>] and [-nofilter <f>] arguments as you would like to specify. Valid filters, <f>, exactly match the set of words that can be applied to the WildcardFilter. The command uses the WildcardFilter first, then applies any user specified filters. The -filter values are added to the filter, the -nofilter values are removed from the filter. Filters are applied in the order specified, so any conflicts are resolved with the last specified wins.
- **-flush [<object>]**
(optional) Forces 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. Specifying <object> logs that object, and then flushes the file.
- **-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 (such as log -r /*). By default, normal wildcard usage sometimes excludes Questa Verification IP transactions from logging.

- **-mvcovm**
(optional) Specifies the inclusion of all UVM and/or OVM sequence transactions when a wildcard is used (such as `log -r /*`). By default, normal wildcard usage sometimes excludes UVM/OVM transactions from logging.
 - **-mvcrecccomplete**
(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. Omitting `-recursive` limits the search 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 to log. Must be the final argument to the **log** command. Specify multiple object names as a space-separated list. Allows wildcard characters. 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

[dataset alias](#)
[dataset clear](#)
[dataset close](#)
[dataset config](#)
[dataset info](#)
[dataset list](#)
[dataset open](#)
[dataset restart](#)
[dataset rename](#)
[dataset save](#)
[dataset snapshot](#)
[nolog](#)

[Recording Simulation Results With Datasets \[Questa SIM User's Manual\]](#)

[WLFFilename \[Questa SIM User's Manual\]](#)

lshift

Takes a Tcl list as an argument, and shifts it one place to the left, eliminating the left-most element.

Syntax

`lshift <list> [<amount>]`

Description

You can specify the number of shift places to be a number greater than one. Returns nothing.

Arguments

- `<list>`
(required) Specifies the Tcl list to target with lshift. Must be the first argument to the lshift command.
- `<amount>`
(optional) Specifies the number of places to shift. The default is 1.

Examples

```
proc myfunc args {  
    # throws away the first two arguments  
    lshift args 2  
    ...  
}
```

I sublist

Returns a sublist of the specified Tcl list that matches the specified Tcl glob pattern.

Syntax

I sublist <list> <pattern>

Arguments

- <list>
(required) Specifies the Tcl list to target with I sublist. Must be the first argument.
- <pattern>
(required) Specifies the pattern to match within the <list> using Tcl glob-style matching.
Must be the final argument.

Examples

- In the example below, variable 't' returns "structure signals source".

```
set window_names "structure signals variables process source wave list"  
set t [I sublist $window_names s*]
```

mem compare

Compares a selected memory to a reference memory or file.

Note

 The "diff" utility must be installed and be visible in your search path in order to run this command.

Syntax

```
mem compare {[-mem <ref_mem>] | [-file <ref_file>] } [actual_mem]
```

Arguments

- **-mem <ref_mem>**
(optional) Specifies a reference memory to compare with actual_mem.
 <ref_mem> — A memory record.
- **-file <ref_file>**
(optional) Specifies a reference file to compare with actual_mem.
 <ref_file> — A saved memory file.
- **actual_mem**
(required) Specifies the name of the memory to compare against the reference data. Must be the final argument to the mem compare command.

mem display

Prints the memory contents of the specified instance to the Transcript window.

Note

 You do not need to specify the signal or variable name if the given instance path contains only a single array signal or variable.

Syntax

```
mem display [-addressradix [d | h]] [-compress] [-dataradix <radix_type>]  
[-endaddress <end>][-format [bin | hex | mti]] [-noaddress] [-startaddress <st>]  
[-wordsperline <n>] [<path>]
```

Description

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 you specify Verilog memory format (hex or binary).

You can specify the address radix, data radix, and address range for the output, 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

- **-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 to not print identical lines. 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 you do not specify a radix for an enumerated type, the symbolic representation is used.

You can use the [radix](#) command to change the default radix type for the current simulation, or edit the DefaultRadix variable in the modelsim.ini file to make the default radix permanent. Refer to [DefaultRadix](#) in the User's Manual for more information.

- **-endaddress <end>**
(optional) Specifies the end address for a range of addresses to display.
 <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 not to print addresses.
- **-startaddress <st>**
(optional) Specifies the start address for a range of addresses to display.
 <st> — Any valid address in the memory. If unspecified, the default is the start of the memory.
- **-wordsperline <n>**
(optional) Specifies how many words to print 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. Must be the final argument to the mem display command. The default is the current context, as shown in the Structure window. You can specify indexes.

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

[mem load](#)

mem list

Displays a flattened list of all memory instances in the current or specified context after a design has been elaborated.

Syntax

mem list [-r] [<path>]

Description

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.

Arguments

- **-r**
(optional) Specifies to recursively descend into sub-modules when listing memories.
- **<path>**
(optional) Specifies 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

Updates the simulation memory contents of a specified instance.

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]
```

Description

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 you do not specify an address range (starting and ending address), the memory is loaded starting at the first location.

You can upload contents either from a memory data file, a memory pattern, or both. If you specify both, the pattern is applied only to memory locations not contained in the file.

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 to 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 depends on the -filltype specified:

- For fixed pattern values, the fill pattern is repeatedly tiled to initialize the specified memory block. 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 is 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. You can specify a seed value on the command line. For any given seed, the generated sequence is identical.

The pattern data is interpreted according to the default system radix setting. However, you can override this setting with a standard Verilog-style '<radix_char><data>' specification.

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 load.
 <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. Specifying a fill pattern without a file option fills the entire memory or specified address range with the specified pattern. If you specify both, the pattern applies 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 the data word to use to fill memory addresses in the pattern specified by -filltype.
 <data_word> — Specifies the data word. Must be in the same format specified by the -fillradix switch.
- **-fillradix <radix_type>**
(optional, use with -filldata) Specifies the 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 load.
 - bin — Specifies binary data format.
 - hex — Specifies hex 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.

The MTI memory data file format stores internal file address and data radix settings within the file itself, so you do not 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, you can use shorthand instance names. The default is the current context, as shown in the Structure window.

You can specify memory address indexes 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 skip 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 load.

<st> — Any valid address in the memory.

- -truncate

(optional) Ignores any most significant bits (msb) in a memory word that 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: 000000000000100 000000000000101 000000000000110
000000000000111
# 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

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.

Syntax

```
mem save -outfile <filename> [-addressradix {dec | hex}] [-dataradix <radix_type>]  
[-format {bin | hex | mti}] [-compress | -noaddress] [<path>]  
[-startaddress <st> -endaddress <end>] [-wordsperline <Nwords>]
```

Description

This command works identically to the mem display command, except that it writes output to a file rather than a display.

The -format argument specifies the order in which data is written to the file. 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.

Arguments

- **-outfile <filename>**
(required) Specifies to store the memory contents in a file.

 <filename> — The name of the file in which to store the specified memory contents.
- **-addressradix {dec | hex}**
(optional) Specifies the address radix for the default mti formatted files.

 dec — Decimal (default).
 hex — Hexadecimal.
- **-compress**
(optional) Specifies to print only unique lines, and not identical lines. 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 use the [radix](#) command to change the default radix for the current simulation. You can edit the DefaultRadix variable in the modelsim.ini file to change the default radix permanently . Refer to [DefaultRadix](#) in the User's Manual for more information.

- **-endaddress <end>**

(optional) Specifies the end address for a range of addresses to save.

<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. You can use these formats with Verilog memories, and with VHDL memories composed of std_logic types.

The MTI memory data file format stores internal file address and data radix settings 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 save.

<st> — Any valid address in the memory.

- **-wordsperline <Nwords>**

(optional) Specifies how many memory values to print 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 00000000000000010
00000000000000011
4: 00000000000000100 0000000000000101 0000000000000110
0000000000000111
8: 0000000000001000 0000000000001001 xxxxxxxxxxxxxxxxx
```

Related Topics

[mem display](#)

[mem load](#)

mem search

Finds and prints to the screen the first occurring match, or all matches, of a specified memory pattern in the specified memory instance.

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. Specify multiple word patterns as a space separated list.
Accepts wildcards.

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. Accepts wildcards. Specify multiple word patterns 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. The default is to print only the first matching occurrence.

- **-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 use the [radix](#) command to change the default radix for the current simulation. You can change the default radix permanently by editing the DefaultRadix variable in the modelsim.ini file. Refer to [DefaultRadix](#) in the User's Manual for more information.

- **-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. Specify multiple word patterns as a space-separated list. 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 -addressradix 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>
```

msgZeroMessageCounts

Resets message counts to zero.

Usage

`msgZeroMessageCounts`

Arguments

None

Description

At 'restore' time, the message counts are set to the values they had at 'checkpoint' time. After the 'restore', you can use the 'msgZeroMessageCounts' command to reset the message counts to 0.

Chapter 3

Commands (N - Z)

This chapter describes Questa SIM commands, listed alphabetically from N through Z, that you can enter either on the command line of the Main window or in a DO file.

[Supported Commands N-Z](#) continues the listing of commands.

Table 3-1. Supported Commands N-Z

Command name	Action	-batch	-c
next	This command continues a search after you have invoked the search command.	N	N
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.	Y	Y
nolog	This command suspends writing of data to the wave log format (WLF) file for the specified signals.	Y	Y
notepad	This command opens a simple text editor. It may be used to view and edit ASCII files or create new files.	N	N
noview	This command closes a window in the ModelSim GUI. To open a window, use the view command.	N	N
nowhen	This command deactivates selected when commands.	Y	Y
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.	Y	Y
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.	Y	Y
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.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	Y	Y
pa autotestplan	This command creates a test plan file based on coverage information in your power aware simulation.	Y	Y
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.	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. Refer to the section “Generating Reports for Power Aware” in the Power Aware Simulation User’s Manual for more information.	Y	Y
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.	Y	Y
pop	This command moves the specified number of call frames up the C callstack. This command is used with C Debug.	N	Y
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.	Y	Y
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.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	Y	Y
power report	This command reports power information for the objects specified with the power add command.	Y	Y
power reset	This command selectively resets power information to zero for the signals or nets specified with the power add command.	Y	Y
precision	This command determines how real numbers display in the graphic interface (for example, 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.	Y	Y
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.	Y	Y
process report	This command creates a textual report of all processes displayed in the Process Window.	Y	Y
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.	Y	Y
profile interval	This command selects the frequency with which the profiler collects samples during a run command.	Y	Y
profile off	This command disables runtime memory allocation and statistical performance profiling.	Y	Y
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.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	Y	Y
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.	Y	Y
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.	Y	Y
profile report	This command outputs profiling data that have been gathered up to the point that you execute the command.	Y	Y
profile save	This command saves the profile data to an external database when ModelSim shuts down. This command is useful for sending profile information to another user for analysis.	Y	Y
profile summary	This command outputs profiling data that is broken into categories based on context in which the callstacks are found.	N	Y
project	This command is used to perform common operations on projects.	N	Y
property list	This command changes one or more properties of the specified signal, net, or register in the List window.	N	N
property wave	This command changes one or more properties of the specified signal, net, or register in the Wave window.	N	N

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
push	This command moves the specified number of call frames down the C callstack.	N	Y
pwd	This Tcl command displays the current directory path in the Transcript window.	Y	Y
questasim	This command starts the QuestaSim GUI without prompting you to load a design.	Y	Y
quietly	This command turns off transcript echoing for the specified command.	Y	Y
quit	This command exits the simulator.	Y	Y
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 ModelSim.	N	Y
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.	Y	Y
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.	Y	Y
radix delete	This command will remove the radix definition from the named radix.	Y	Y
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.	Y	Y
radix names	This command returns a list of currently defined radix names.	Y	Y
radix signal	This command sets or inspects radix values for the specified signal in the Objects, Locals, Schematic, and Wave windows. When no argument is used, the radix signal command returns a list of all signals with a radix.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
readers	This command displays the names of all readers of the specified object.	N	Y
report	This command displays information relevant to the current simulation.	Y	Y
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.)	Y	Y
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").	Y	Y
resume	This command is used to resume execution of a macro (DO) file after a pause command or a breakpoint.	Y	Y
right	This command searches right (next) for signal transitions or values in the specified Wave window.	N	N
run	This command advances the simulation by the specified number of timesteps.	Y	Y
runStatus	This command returns the current state of your simulation to stdout after issuing a run or step command.	Y	Y
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.	Y	Y
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.	Y	Y
 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.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	N	N
searchlog	This command searches one or more of the currently open logfiles for a specified condition.	N	Y
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.	Y	Y
seetime	This command scrolls the List or Wave window to make the specified time visible.	N	Y
setenv	This command changes or reports the current value of an environment variable. The setting is valid only for the current ModelSim session. Arguments to this command are order dependent. Please read the argument descriptions for more information.	Y	Y
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.	Y	Y
show	This command lists HDL objects and subregions visible from the current environment.	Y	Y
simstats	This command returns performance-related statistics about elaboration and simulation. The statistics measure the simulation kernel process 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.	Y	Y
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 a list instead of on separate lines.	Y	Y
stack down	This command moves down the call stack.	Y	Y
stack frame	This command selects the specified call frame.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
stack level	This command reports the current call frame number.	Y	Y
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.	Y	Y
stack up	This command moves up the call stack.	Y	Y
status	This command lists summary information about currently interrupted macros.	Y	Y
step	The step command is an alias for the run command with the -step switch. Steps the simulator to the next HDL.	Y	Y
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.	Y	Y
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.	Y	Y
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.	Y	Y
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.	Y	Y
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.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
Time	These commands allow you to perform comparisons between, operations on, and conversions of, time values.	Y	Y
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).	Y	Y
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 - togglename”.	Y	Y
toggle enable	The toggle enable command re-enables toggle statistics collection on nodes whose toggle coverage had previously been disabled.	Y	Y
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.	Y	Y
toggle reset	The toggle reset command resets the toggle counts to zero for the specified nodes.	Y	Y
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.	N	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	N	Y
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.	N	Y
transcribe	This command displays a command in the Transcript window, and then executes the command.	N	Y
transcript	This command controls echoing of commands executed in a macro file. If no option is specified, the current setting is reported.	Y ¹	Y
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.	Y	Y
transcript path	This command returns the full pathname to the current transcript file.	Y	Y
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.	Y	Y
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 UCDB, a WLF file, or a LOG file	N	N
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:	N	N

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	N	N
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.	N	N
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.	N	N
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.	N	N
tssi2mti	This command is used to convert a vector file in TSSI Format into a sequence of force and run commands.	N	Y
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.	N	Y
unsetenv	This command deletes an environment variable. The deletion is not permanent – it is valid only for the current ModelSim session.	Y	Y
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.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
up	This command searches for object transitions or values in the specified List window.	N	N
uvm call	This command calls a SystemVerilog UVM function, either from the uvm_pkg or in a user's UVM test bench code.	Y	Y
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.	Y	Y
uvm displayobjections	This command returns the current objections to the transcript. If no argument is supplied, display objections for uvm_top.	Y	Y
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.	Y	Y
uvm findsequences	This command returns the active sequences under a specified sequencer. Returns all active sequences when specified without arguments.	Y	Y
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.	Y	Y
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.	Y	Y
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.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
uvm printfactory	This command returns UVM global factory information to the transcript. This command does not take any arguments	Y	Y
uvm printstreams	This command returns a list of streams (transaction objects) to the transcript.	Y	Y
uvm printtopology	This command returns the UVM global testbench topology to the transcript. This command does not take any arguments.	Y	Y
uvm setverbosity	This command sets the global reporting verbosity.	Y	Y
uvm simpAth	This command maps the path of a UVM hierarchical component to a simulator "/uvm_root" context hierarchy string.	Y	Y
uvm traceobjections	This command enables or disables objections tracing for a specified UVM component instance.	Y	Y
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.	Y	Y
vcd add	This command adds the specified objects to a VCD file.	Y	Y
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	Y	Y
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.	Y	Y
vcd dumports	This command creates a VCD file that includes port driver data.	Y	Y
vcd dumportsall	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: \$dumportsall	Y	Y
vcd dumportsflush	This command flushes the contents of the VCD file buffer to the specified VCD file. Related Verilog task: \$dumportsflush	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	Y	Y
vcd dumpportoff	This command turns off VCD dumping and records all dumped port values as x.	Y	Y
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 dumpportoff. Related Verilog task: \$dumpportson	Y	Y
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.	Y	Y
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	Y	Y
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	Y	Y
vcd limit	This command specifies the maximum size of a VCD file (by default, limited to available disk space).	Y	Y
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	Y	Y
vcd on	This command turns on VCD dumping to the specified file and records the current values of all VCD variables.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
vcd2wlf	This command is a utility that translates a VCD (Value Change Dump) file into a WLF file that you can display in ModelSim using the vsim -view argument. This command only works on VCD files containing positive time values.	Y	Y
vcom	The vcom command compiles VHDL source code into a specified working library (or to the work library by default).	Y	Y
vcover attribute	This command is used to display attributes in the currently loaded database during batch mode simulation.	Y	Y
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.	Y	Y
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.	Y	Y
vcover history	This command allows you to access 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.	Y	Y
vcover merge	The vcover merge command merges multiple code coverage data files that were created with the coverage save command into a single UCDB output file.	Y	Y
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.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	Y	Y
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.	Y	Y
vcover report	The vcover report command prints textual output of coverage statistics or exclusions — from a previously saved code, coverage run — to a specified file.	Y	Y
vcover stats	The vcover stats command prints to stdout summary statistics for previously saved code coverage databases.	Y	Y
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.	Y	Y
vdel	This command deletes a design unit from a specified library. This command provides additional information with the -help switch.	Y	Y
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 ModelSim.	Y	Y
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.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
<code>verror</code>	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.	Y	Y
<code>vgencomp</code>	Once a Verilog module is compiled into a library, you can use this command to write its equivalent VHDL component declaration to standard output.	Y	Y
<code>vhencrypt</code>	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.	Y	Y
<code>view</code>	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.	N	N
<code>virtual count</code>	This command reports the number of currently defined virtuals that were not read in using a macro file.	N	Y
<code>virtual define</code>	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.	N	Y
<code>virtual delete</code>	This command removes the matching virtuals.	N	Y
<code>virtual describe</code>	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.	N	Y
<code>virtual expand</code>	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.	N	Y
<code>virtual function</code>	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>.	N	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	N	Y
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.	N	Y
virtual nohide	This command reverses the effect of a virtual hide command, causing the specified real or virtual signals to reappear the Objects window.	N	Y
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.	N	Y
virtual region	This command creates a new user-defined design hierarchy region.	N	Y
virtual save	This command saves the definitions of virtuals to a file named virtual.do in the current directory.	N	Y
virtual show	This command lists the full path names of all explicitly defined virtuals.	N	Y
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>.	N	Y
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.	N	Y
vlib	This command creates a design library. You must use vlib rather than operating system commands to create a library directory or index file.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	Y	Y
vmake	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.	Y	Y
vmap	The vmap command defines a mapping between a logical library name and a directory by modifying the modelsim.ini file.	Y	Y
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.	Y	Y
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)	Y	Y
vsim<info>	The vsim<info> commands return information about the current vsim executable.	Y	Y
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.	Y	Y
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..	Y	Y
wave	A number of commands are available to manipulate and report on the Wave window.	N	N

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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.	N	N
wave edit	This command modifies waveforms created with the wave create command.	N	N
wave export	This command creates a stimulus file from waveforms created with the wave create command.	N	N
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 ModelSim. Use this command to apply a VCD file as stimulus to the current simulation.	N	N
wave modify	This command modifies waveform parameters set by a previous wave create command.	N	N
wave sort	This command sorts signals in the Wave window by name or full path name.	N	N
when	This command instructs ModelSim to perform actions when the specified conditions are met.	Y	Y
where	This command displays information about the system environment. It is useful for debugging problems where ModelSim cannot find the required libraries or support files.	Y	Y
wlf2log	This command translates a ModelSim 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.	Y	Y
wlf2vcd	This command translates a ModelSim WLF file to a standard VCD file. Complex data types that are unsupported in the VCD standard (records, memories, and so on) are not converted.	Y	Y
wlfman	This command allows you to get information about and manipulate saved WLF files.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
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 ModelSim> prompt or from a shell.	Y	Y
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.	Y	Y
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.	N	Y
write list	This command records the contents of the List window in a list output file.	N	Y
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.	N	Y
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.	Y	Y
write timing	This command displays path delays and timing check limits, unadjusted for delay net delays, for the specified instance.	Y	Y
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).	N	Y
write tssi	This command records the contents of the List window in a "TSSI format" file.	Y	Y

Table 3-1. Supported Commands N-Z (cont.)

Command name	Action	-batch	-c
write wave	This command records the contents of the Wave window in PostScript format. The output file can then be printed on a PostScript printer.	N	N
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.	Y	Y
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-propagated assertions.	Y	Y
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.	Y	Y
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.	Y	Y

1. transcript on | off only are supported.

next

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.

noforce

Removes the effect of any active force commands on the selected HDL objects, and also causes the object's value to be re-evaluated.

Syntax

noforce <object_name> ...

Description

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.

Arguments

- <object_name>

(required) Specifies the name of an object. Must match an object name used in a previous [force](#) command. You can specify multiple object names as a space-separated list. Allows wildcard characters.

Related Topics

[force](#)

nolog

Suspends writing of data to the wave log format (WLF) file for the specified signals.

Syntax

```
nolog [-all] [-depth <level>] [-howmany] [-in] [-inout] [-internal] [-out] [-ports] [-recursive]  
[-reset] [<object_name>...]
```

Description

Writes a flag 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. You can turn logging back on by issuing another [log](#) command or by doing a nolog -reset.

Because the nolog command adds new information to the WLF file, older versions of the simulator cannot read WLF files created when using the nolog command.

Transactions written in SCV or Verilog are logged automatically, and you can remove them with the nolog command. A transaction is logged into the .wlf file if logging is enabled for that stream, and hasn't been disabled with nolog, when the transaction begins. The entire span of a transaction is either logged or not logged, regardless of the begin and end times specified for that transaction.

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. Omitting -recursive limits the search 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 to unlog. You can specify multiple object names as a space-separated list. Allows wildcard characters.

Examples

- Unlog all objects in the design.
nolog -r /*
- Turn logging back on for all unlogged signals.
nolog -reset

Related Topics

[log](#)

notepad

Opens a simple text editor to view and edit ASCII files or create new files.

Syntax

`notepad [<filename>] [-r | -edit]`

Description

You can change this mode from the Notepad Edit menu.

Returns nothing.

Arguments

- <filename>
(optional) Name of the file to display.
- -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

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. You can specify multiple window types in a space-separated list. Allows wildcards. Requires at least one type (or wildcard).

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

nowhen

Deactivates selected when commands.

Syntax

nowhen [<label>]

Arguments

- <label>

(optional) Specifies an individual when command. You can use wildcards to select more than one when command.

Examples

- Deactivate the **when** command labeled 99.

nowhen 99

- Deactivate all **when** commands.

nowhen *

onbreak

Used in a DO file to specify one or more scripts to execute when the simulation encounters a breakpoint in the source code.

Syntax

```
onbreak <script>; <script>...
```

Description

You must include a [run](#) or [step](#) command after any `onbreak` command in your DO file.

An `onbreak` script affects any subsequent run commands, until another `onbreak` command is issued. If a script executes a DO file, the contents of the DO file inherit the calling script's `onbreak` setting, unless and until the DO file executes another `onbreak` command. In that case, the new `onbreak` setting takes effect until the DO file script completes. Execution then returns to the calling script, restoring the calling script's `onbreak` setting. (Refer to [Breakpoint Flow Control in Nested DO files](#) in the User's Manual for more information.)

The `OnBreakDefaultAction` Tcl variable defines the default behavior when there is no `onbreak` setting in effect. If the `OnBreakDefaultAction` variable is not defined, the simulator default is to resume execution.

You can use an empty string to change the `onbreak` command back to the default behavior:

```
onbreak ""
```

In this case, the script resumes after a breakpoint occurs (after any associated [bp](#) command string is executed).

If you do not specify an argument to the `onbreak` command, it returns either the definition of any scripts defined by previous `onbreak` commands, or nothing, if you have not previously defined any `onbreak` commands.

Arguments

- `<script>`

(optional) Any command or script can be used as an argument to `onbreak`. To use more than one command or script, separate them by semicolons, or place them on multiple lines and enclose the entire script in braces (`{ }`) or quotation marks (`" "`). If a `run` or `step` command is issued within an `onbreak` script, the script returns immediately, and without executing any of the following commands. It is an error to execute any commands following a `run` command within an `onbreak` command string. 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](#) in the User's Manual). 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[do](#)[onerror](#)[Useful Commands for Handling Breakpoints and Errors \[Questa SIM User's Manual\]](#)[DO Files \[Questa SIM User's Manual\]](#)

onElabError

Used in a DO file script, specifies one or more commands to execute when an error is encountered during the elaboration portion of a vsim command.

Syntax

```
onElabError {[<command> [<command>] ...]}
```

Arguments

- <command>

(optional) Any command can be used as an argument. If you use more than one command, separate them with semicolons, or place them on multiple lines. You must enclose the entire command string in braces ({}). If you do not enter any arguments, onElabError returns to a prompt.

Related Topics

[do](#)

onerror

Used in a DO file before a run command, specifies one or more commands to execute when a running script encounters an error.

Syntax

```
onerror {[<command> [; <command>] ...]}
```

Description

Using the onerror command without arguments returns 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](#)” in the GUI Reference Manual for details).

With a successful onerror command, the DO file script continues normally, unless the command instructs the simulator to quit, as for example:

```
onerror{quit -f}
```

or

```
onerror {break}
```

An unsuccessful onerror command causes the simulator to quit. For example:

```
onerror {add wave b}
```

when you do not have a signal named b.

The onerror command executes when a Tcl command (such as 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 runs even if the DO file does not contain a local onerror command, which is useful when you run a series of DO files from one script, and want the same behavior across all DO files.

Arguments

- <command>

(optional) Specifies the command or commands to run. Separate multiple commands with semicolons, or place them on multiple lines. You must enclose the entire command string in braces ({}).

Examples

- Force the simulator to quit if an error is encountered while the DO file is running.

onerror {quit -f}

Related Topics

[do](#)

[onbreak](#)

[Useful Commands for Handling Breakpoints and Errors \[Questa SIM User's Manual\]](#)

[DO Files \[Questa SIM User's Manual\]](#)

onfinish

Controls simulator behavior when encountering \$finish or sc_stop() in the design code.

Note

 Specifying this command without an argument 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, to simplify post-simulation tasks.
- `default`
(optional) Uses the current setting for the OnFinish variable in the *modelsim.ini* file.

Related Topics

[OnFinish \[Questa SIM User's Manual\]](#)

pa autotestplan

Creates a testplan file based on coverage information in your Power Aware simulation.

Syntax

```
pa autotestplan [-format={xml | ucdb}] [-filename=<filename>]
```

Description

Refer to “[Analyzing Coverage Using the Power Aware Test Plan](#)” in the *Power Aware Simulation User’s Manual* for more information.

Arguments

- `-format={xml | ucdb}`

Specifies the filetype format to write. The default is UCDB format.

If you generate the testplan in XML format, you can use the [xml2ucdb](#) command to convert to UCDB format.

- `-filename=<filename>`

Specifies the filename of the testplan. The default is *QuestaPowerAwareTestplan.ucdb*.

pa msg

Provides granular control of Power Aware dynamic checks, and controls the display of warning and error messages that occur during these checks.

Note

 Enabling granular control of dynamic checks also enables the underlying assertions, including warning and error messages, and any resulting coverage collection that occurs.

You can run this command at the VSIM prompt, or include it in a Tcl configuration file and invoke the file at the VSIM prompt.

Syntax

To select dynamic checks

```
pa msg [-checkIds <list-of-dyn-chksId>] [<message numbers>] [-pa_checks=[<spec>]] [-all]
```

To granularly control the selected dynamic checks

```
pa msg [-domains <list-of-domain-name>] [-domain <pd_name>] [-strategies <list-of-strategy-name>] [-strategy <strategy_name>] [-powerswitches <list-of-switch-name>] [-elements <list-of-elements>] [-elements <list-of-elements>] [-ports <list_of_ports>] [-pst <list-of-pst-names>] [-transitive [<TRUE | FALSE>]] [-models <list-of-modules>] [-scope <scope>]
```

To take action on the selected dynamic checks

```
pa msg [-enable | -disable] [-severity {note | warning | error | fatal}] [-outfile <pa-assertion-outfile-name>] [-glitch_window <duration><time_spec>] [-stopafter <number>] [-suppressafter <number>]
```

Arguments

Note

 If you do not specify a dynamic check, the command selects all dynamic checks, except the ones with the severity NOTE.

- **-checkIds <list-of-dyn-chksId>**

(optional) Specifies a list of one or more dynamic checks to enable, disable, set the severity for, and so on.

The values you specify in <list-of-dyn-chksId> can be any of the following types:

- The argument values of the vopt -pa_checks command (such as ira, irc, ifc).
- The mnemonics of checks flagged in the error message (such as QPA_UPF_INCORRECT_LS_CHK, QPA_UPF_MISSING_LS_CHK).
- The numerical value of the error messages (such as 8915, 8930).

If you specify a value in <list-of-dyn-chksId> that does not exist, a semantic error results.

- <message numbers>

(optional) Specifies a space-separated list of one or more dynamic checks that you want to enable, disable, set the severity for, and so on. Specify the numerical values of the dynamic check messages in <message numbers>. If you do not specify any message numbers, the arguments of the pa msg command affect all dynamic checks. If you specify a value in <message numbers> that does not exist, a semantic error results.

Note

 The <message numbers> argument is deprecated. Use the -checkIds argument instead.

- -pa_checks=[<spec>]

(optional) Specifies a list of one or more dynamic checks to enable, disable, set the severity for, and so on. Specify the argument values of the vopt -pa_checks command (such as icp, ifc, ira) in <spec>. To specify multiple dynamic checks, use a plus sign (+) between values. If you do not specify any dynamic checks, then the arguments of the pa msg command affect all dynamic checks.

Note

 The -pa_checks=<spec> argument is deprecated. Use the -checkIds argument instead.

- -all

(optional) Selects all dynamic checks.

- -domains <list-of-domain-name>

(optional) Restricts the selected dynamic check to the power domains listed in <list-of-domain-name> and their UPF objects (such as isolation, level shifter, retention strategies). Does not affect messages related to checks that are not associated with any power domain, such as uml and umi. If you do not specify -domains, the selected dynamic check is applied to all power domains. If you specify a power domain that does not exist, a warning message is issued, and the argument is ignored.

- -domain <pd_name>

(optional) Restricts the selected dynamic check to the power domain listed in <pd_name> and its UPF objects (such as isolation, level shifter, retention strategies). Does not affect messages related to checks that are not associated with any power domain, such as uml and umi. If you do not specify -domain, the selected dynamic check is applied to all power domains. If you specify a power domain that does not exist, a warning message is issued, and the argument is ignored.

Note

 The -domain argument is deprecated. Use the -domains argument instead.

- **-strategies <list-of-strategy-name>**

(optional) Restricts the selected dynamic check to the strategies listed in <list-of-strategy-name>. Does not affect checks that are not associated with any strategy, such as p, t, uml, umi, and npu. If you do not specify -strategies, the selected dynamic check is applied to all strategies. If you specify a strategy that does not exist, a warning message is issued, and the argument is ignored.

Use the -strategies argument with the -domain argument, to select dynamic checks in domain.strategy.

- **-strategy <strategy_name>**

(optional) Restricts the selected dynamic check to the strategy listed in <strategy_name>. This argument has no affect on checks that are not associated with any strategy, such as p, t, uml, umi, and npu. If you do not specify -strategy, the selected dynamic check is applied to all strategies. If you specify a strategy that does not exist, a warning message is issued, and the argument is ignored.

Note



The -strategy argument is deprecated. Use the -strategies argument instead.

- **-powerswitches <list-of-switch-name>**

(optional) Restricts the selected dynamic check to the power switches listed in <list-of-switch-name>. If you do not specify -powerswitches, the selected dynamic check is applied to all power switches. If you specify a power switch that does not exist, a warning message is issued, and the argument is ignored.

- **-elements <list-of-elements>**

(optional) Restricts the selected dynamic check to the design elements (such as instance, port, net or signal names) listed in <list-of-elements>. If you do not specify -elements, the selected dynamic check is applied to all elements. If you specify an element that does not exist, a warning message is issued, and the argument is ignored.

The -elements argument is not applicable to all dynamic checks. You receive a NOTE message when the -elements argument is not applicable to a dynamic check, and the argument is ignored.

- **-ports <list_of_ports>**

(optional) Restricts the selected dynamic check to the ports listed in <list_of_ports>. If you do not specify -ports, the selected dynamic check is applied to all ports. If you specify a port that does not exist, a warning message is issued, and the argument is ignored.

The -ports argument is not applicable to all dynamic checks. You receive a NOTE message when the -ports argument is not applicable to a dynamic check, and the argument is ignored.

Note



The -ports argument is deprecated. Use the -elements argument instead.

- **-pst**s <list-of-pst-names>
(optional) Restricts the selected dynamic check to the power state tables (PSTs) listed in <list-of-pst-names>. If you do not specify -pst, the selected dynamic check is applied to all PSTs. If you specify a PST that does not exist, a warning message is issued, and the argument is ignored.
- **-transitive** [<TRUE | FALSE>]
(optional) Determines the top-level hierarchy at which the dynamic check is performed. For use only in contexts where instance hierarchies are specified with the -elements argument. The -transitive argument has no meaning if you specify it without the -elements argument.
 - TRUE (default) — Specifies that the pa msg command applies to all hierarchies on and below the specified instance path.
 - FALSE — Specifies that the pa msg command applies only to the specified design hierarchies that have instance names specified by the -elements argument.
- **-models** <list-of-modules>
(optional) Restricts the selected dynamic check to a list of Verilog modules or VHDL entities specified in <list-of-modules>. If you do not specify -models, the selected dynamic check is applied to all models. If you specify a model that does not exist, a warning message is issued, and the argument is ignored.
- **-scope** <scope>

Note

You must accompany the -scope argument with a -domains, -elements, -models, -ports, or -strategies argument.

(optional) Restricts the selected dynamic check to the scope specified by <scope>. The pa msg command applies the selected dynamic check to the specified domains/strategies/ports/elements/models, and their UPF objects (such as isolation, level shifter, retention strategies).

- **-enable | -disable**
(optional) Enables or disables the selected dynamic check. Enabling the dynamic check also enables the underlying assertions, including the warning and error messages, and any resulting coverage collection that occurs during dynamic check. If you do not use either the -enable or the -disable argument with a pa msg command, then -enable is applied by default.

Caution

Attempting to use the -enable and -disable arguments on the same check causes a syntax error.

- **-enable**
Enables the selected dynamic checks.
- **-disable**

Disables the selected dynamic checks.

- **-severity {note | warning | error | fatal}**

(optional) Sets the severity of the messages of the selected dynamic check. If you do not select any dynamic check, then the **-severity** argument applies to all dynamic check messages.

- **-outfile <pa-assertion-outfile-name>**

(optional) Redirects the messages resulting from the selected dynamic check to a specified text file. If you do not select any dynamic check, this argument applies to all dynamic check messages. By default, the messages display in the transcript window.

- **-glitch_window <duration><time_spec>**

(optional) Specifies the maximum allowed time window of a glitch on the control lines. A spike is treated as a glitch if its duration is greater than the maximum allowed time window, otherwise the spike is ignored. This argument applies to glitch detection checks (-pa_checks=ugc) only. If specified with any other checks, this argument is ignored.

<duration> — Specifies the maximum allowed time window of a glitch. You can specify any positive real number or zero (0). The default value is zero.

<time_spec> — Specifies the time unit of **<duration>**. It can be fs, ps, ns, us, or ms. The default value is the time unit of your simulation.

Note

 There is no space between **<duration>** and **<time_spec>**.

- **-stopafter <number>**

(optional) Stops the simulation after a selected dynamic check message(s) is displayed for **<number>** times. If you do not select a specific dynamic check, the simulation stops when any dynamic check message is displayed for **<number>** times.

- **-suppressafter <number>**

(optional) Limits the number of message(s) of a selected dynamic check to **<number>**, where the default value is 15. Once the limit is reached, the selected dynamic check messages are suppressed. If you do not select a specific dynamic check, then Questa SIM limits the number of messages of all dynamic checks to **<number>**.

Note

 Not all arguments are applicable to all Power Aware dynamic checks. Questa SIM ignores arguments when they are not applicable to a Power Aware dynamic check.

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

[Command to Control Dynamic Checks \[Power Aware Simulation User's Manual\]](#)

pa report

Generates all Power Aware reports in the *pa_reports* directory within the current working directory.

Syntax

`pa report`

Description

Before you can run `pa report`, you must:

1. Run the `vopt` command with the `-pa_genrpt` argument to collect the necessary information for the report.
(optional) Use the `-pa_reportdir` argument with the `vopt` command to change the default location of the reports.
2. Load the optimized design for simulation.

You can then run the `pa report` command at the `vsim` prompt.

Arguments

None

Related Topics

[Generating Reports for Power Aware \[Power Aware Simulation User's Manual\]](#)

pause

Inserted in a macro, interrupts execution of the macro, allowing you to perform interactive debugging.

Syntax

```
pause
```

Description

When a macro is interrupted during execution, the macro returns the prompt:

```
VSIM(paused) >
```

to notify 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 invoke a third macro, and you can continue invoking macros up to a nesting level of 50 macros.

Use the [status](#) command to list summary information about all interrupted macros.

Use the [resume](#) command to resume execution of a macro.

Use the [abort](#) command to stop execution of some or all of the macros.

Arguments

None.

Related Topics

[resume](#)

[run](#)

[status](#)

pop

Used with C Debug, moves the specified number of call frames up the C callstack.

Syntax

pop <#_of_levels>

Arguments

- **<#_of_levels>**

(optional) Specifies the number of call frames to move up the C callstack. If unspecified, moves up one level.

Examples

- Move up 1 call frame.

pop

- Move up 4 call frames.

pop 4

Related Topics

[push](#)

[C Debug \[Questa SIM User's Manual\]](#)

power add

Specifies the signals or nets to monitor for power information.

Syntax

```
power add [-duplicates] [[-in] [-out] [-inout] | [-ports]] [-internal] [-nocellnet] [-noreg] [-r]
           [-vhdlint] <signals_nets> ...
```

Description

When you define power add for a signal or net, the simulator keeps a record of any switching activity of that signal. The simulator writes this information to a file when you [run the power report command](#). You can translate and use this data with third-party power analysis tools.

Note

 You can use the [power off](#) command to disable monitoring between runs and then use the [power on](#) command to resume monitoring.

Arguments

- **-duplicates**
(optional) Dumps the switching activity of duplicate signals in each hierarchy.
Duplicate signals are connected nets or ports that have the same switching activity values. For example, a signal that is passing through multiple hierarchical boundaries.
- **-in**
(optional) *For use with wildcard searches.* Specifies that the scope of the search includes 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 includes 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 includes 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 includes 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 includes 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.
- **-noreg**
(optional) Disables the output of switching activity of design elements inferred as registers, such as reg, logic, bit, and function.
- **-r**
(optional) Searches recursively on a wildcard specified for the signal or net.
- **-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), a logical OR of the arguments is performed.

- **<signals_nets> ...**
(required) Specifies the signal or net path to monitor. Must be 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; one- and two-dimensional arrays (packed and unpacked); reg, logic and bit datatypes; and structs and arrays (up to 3-dimensional) of reg, logic and bit datatype.

When using wildcards, the -in, -inout, -internal, -out, and -ports arguments filter the qualifying signals.

Related Topics

- [power off](#)
- [power on](#)
- [power report](#)
- [power reset](#)

power off

Works in conjunction with the power add command to stop vsim from updating toggle activity data for the specified signal or net.

Syntax

```
power off [-all] [[-in] [-out] [-inout] | [-ports]] [-internal] [-r] <signals_nets> ...
```

Description

When this command is executed, subsequent run commands ignore the power add command for the specified arguments.

Arguments

- **-all**
(optional) For use with wildcard searches. Specifies that the scope of the search includes 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 includes 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 includes 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 includes 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 includes 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 includes 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 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; one- and two-dimensional arrays (packed and unpacked); reg, logic and bit datatypes; and structs and arrays (up to 3-dimensional) of reg, logic and bit datatype.

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 on](#)

[power report](#)

[power reset](#)

power on

Works in conjunction with the power add command to cause vsim to begin or resume updating toggle activity data for the specified signal or net.

Syntax

```
power on [-all] [[-in] [-out] [-inout] | [-ports]] [-internal] [-r] <signals_nets> ...
```

Description

After this command executes, every subsequent run command implements the power add command.

Arguments

- **-all**
(optional) For use with wildcard searches. Specifies that the scope of the search includes 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 includes 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 includes 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 includes ports of mode INOUT, if they match the <signal_nets> specification.
- **-ports**
(optional) For use with wildcard searches. Specifies that the scope of the search includes 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 includes 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 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; one- and two-dimensional arrays (packed and unpacked); reg, logic and bit datatypes; and structs and arrays (up to 3-dimensional) of reg, logic and bit datatype.

When using wildcards, the -in, -inout, -internal, -out, and -ports arguments filter the qualifying signals.

If you specify more than one argument (-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

Reports power information for the objects specified with the power add command.

Syntax

```
power report [-all] [-noheader] [-file <filename>] [-bsaif <filename>]
```

Description

To use the power report command, you must first add the objects of interest with the [power add](#) command, then run the simulation with the [run](#) command.

The report can be in either a tabular ASCII format (-file) or a Switching Activity Interchange Format (SAIF) format (-bsaif). You can translate and use the report data with third-party power analysis tools.

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 transitions where 0->1, 0->x->1, 1->0, or 1->x->0.
- Ti — (hazard count) The number of transitions where 0->x->0 and 1->x->1.

Note

 Note that if a signal is initialized at X, and later transitions to 0 or 1, it is not counted as a hazard.

The UPF standard (IEEE Std 1801-2015) defines inertial glitches as:

...signal transitions occurring at the output of the gate, which can be filtered out if an inertial delay algorithm is applied.

This means 0->x->0 and 1->x->1 transitions are treated as an inertial glitch or hazard, which increments Ti. However, 0->x->1 or 1->x->0 transitions (along with 0->1 and 1->0 transitions) are considered as a valid toggle, which increments Tc.

- Time at X — The length of time spent at each of the three respective states.

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.
 <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 to generate both reports.
- **-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 to generate both reports.

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)

```
(SAIFFILE
(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

Selectively resets power information to zero for the signals or nets specified with the power add command.

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 includes 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 includes 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 includes 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 includes ports of mode INOUT, if they match the <signal_nets> specification.

- **-ports**

(optional) *For use with wildcard searches.* Specifies that the scope of the search includes 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 includes 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 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; one- and two-dimensional arrays (packed and unpacked); reg, logic and bit datatypes; and structs and arrays (up to 3-dimensional) of reg, logic and bit datatype.

When using wildcards, the -in, -inout, -internal, -out, and -ports arguments filter the qualifying signals.

If you specify more than one argument (-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

Determines how real numbers display in the graphic interface (such as the Objects, Wave, Locals, and List windows).

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.

Description

The precision command does not affect the internal representation of a real number and therefore does not allow precision values over 17. Executing the command without any arguments returns the current precision setting.

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 does not print trailing zeros.

precision 8

For example:

1.23456 or 6543.21

printenv

Prints the current names and values of all environment variables to the Transcript window. 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

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 in which to save raw process data. If you do not specify -file, 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 you do not use this option, the process data overwrites the existing process report file.

profile clear

Prerequisites:

Before using this command, do the following:

- Enable profiling with the [profile on](#) command

Clears any performance data that has been gathered during previous executions of the run command, and includes an implicit profile off.

Syntax

`profile clear`

Description

Executing this command resets all profiling data.

This command has no effect on the current profiling session. The last [profile on](#) or [profile off](#) command is still in effect.

Arguments

None

Related Topics

[profile interval](#)

[profile configure](#)

[profile off](#)

[profile on](#)

[profile open](#)

[profile option](#)

[profile reload](#)

[profile report](#)

[profile save](#)

[profile summary](#)

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

profile configure

Enables you to configure what data to query in the profile database.

Usage

```
profile configure [-all {-start <time>} {-end <time>} ] | [-session <name> {-start <time>} {-end <time>} {[ -session ...] }]
```

Description

If -start <time> is not specified, time 0 is used. If -end <time> is not specified, the query results include all samples collected with a time greater than -start <time>.

Arguments

- **-all {-start <time>} {-end <time>}**
(Optional) Resets to the default behavior of using all loaded and collected profile data when generating reports or displaying data.
 - start <time> Reports or displays data beginning at the specified time.
 - end <time> Reports or displays data up to the specified time.
- **-session <name> {-start <time>} {-end <time>}**
Enables you to add a session to the data used for reports. You can provide multiple -session arguments.
 - start <time> Specifies the starting point of the range of reported profile data.
 - end <time> Specifies the ending point of the range of reported profile data.

Examples

Generate a profile calltree report of the combined foo and bar sessions. Report all samples for session foo, but report samples collected only from 100ns to 200ns for session bar.

```
profile configure -session foo -session bar -start 100ns -end 200ns;  
profile report -calltree
```

Related Topics

- [profile clear](#)
- [profile interval](#)
- [profile off](#)
- [profile on](#)
- [profile open](#)
- [profile option](#)
- [profile reload](#)

[profile report](#)

[profile save](#)

[profile summary](#)

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

profile interval

Prerequisites:

Before using this command, do the following:

- Enable profiling by using the [profile on](#) command

Selects the frequency with which the profiler collects samples during a run 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. The default is 10 ms.

Executing the profile interval command without arguments returns the current sample frequency.

Related Topics

[profile clear](#)

[profile configure](#)

[profile off](#)

[profile on](#)

[profile open](#)

[profile option](#)

[profile reload](#)

[profile report](#)

[profile save](#)

[profile summary](#)

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

profile off

Disables statistical performance profiling. A subsequent profile on starts a new session.

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 configure](#)
- [profile interval](#)
- [profile on](#)
- [profile open](#)
- [profile option](#)
- [profile reload](#)
- [profile report](#)

[profile save](#)

[profile summary](#)

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

profile on

Enables runtime memory allocation and statistical performance profiling.

Note

 You cannot save memory allocation profile data and statistical performance data in the same database. Attempting to run memory allocation profiling and statistical performance profiling at the same time causes a non-suppressible error. You must run [profile clear](#) before switching between memory allocation profiling and statistical performance profiling.

Syntax

```
profile on [-p [-name <session>]] [-m [-file <filename> | -fileonly <filename>]]
```

Description

After this command is executed, every subsequent [run](#) command is profiled.

Arguments

- **-m**
(optional) Enables memory allocation profiling only. Deletes any existing memory allocation profile data from a previous invocation of the [profile open](#) command. You cannot run **-m** and **-p** at the same time.
- **-p [-name <session>]**
(optional) Enables statistical performance profiling only. Merges statistical performance profiling data collected from successive simulation runs. You cannot run **-p** and **-m** at the same time.
 - name <session>** (optional) Enables you to supply a name for each statistical performance data session. If you do not provide a name, defaults to the name of the top-level design unit. If an existing session exists with the same name, an integer is appended to the name to make it unique. You can name only a single session during each simulation run.
- **-file <filename>**
(optional) Allows creation of a raw profile data file that can be post-processed later with 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 with 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 configure](#)

[profile interval](#)

[profile off](#)

[profile open](#)

[profile option](#)

[profile reload](#)

[profile report](#)

[profile save](#)

[profile summary](#)

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

profile open

Opens a database containing profile data created with the profile save command.

Syntax

`profile open <filename>`

Arguments

- `<filename>`
(required) An absolute or relative pathname to a file containing the profile database.

Description

Profile open treats memory profile data differently than statistical performance profile data.

- It deletes any existing memory profile data, whether collected during previous activity in simulation mode or from a previous invocation of the profile open command.
- It merges existing statistical performance profile data with data from subsequent simulation runs. It does not merge duplicate data, such as data collected from loading the same database several times. If an existing session exists with the same name as a new session, an integer is appended to the new session name to make it unique.

This command is useful for analyzing profile data from another user.

Related Topics

[profile clear](#)

[profile configure](#)

[profile interval](#)

[profile off](#)

[profile on](#)

[profile option](#)

[profile reload](#)

[profile report](#)

[profile save](#)

[profile summary](#)

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

profile option

Changes how profiling data are reported.

Syntax

profile option collapse_sections [on | off | status]

profile option collect_calltrees [on | off | status]

Description

The profile option command 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

Before using this command, you must enable profiling with the [profile on](#) command.

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 configure](#)

[profile interval](#)

[profile off](#)

[profile on](#)

[profile open](#)

[profile reload](#)

[profile report](#)

[profile save](#)

[profile summary](#)

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

profile reload

Reads in raw profile data from an external file created during memory allocation profiling.

Syntax

profile reload <filename>

Description

The profile reload command clears all performance and memory profiling data collected up to that point. It terminates any currently loaded design (similar to the quit -sim command), and turns off run-time profiling.

You can use the profile report command and the Profile and Profile Details windows of the user interface to view the data. The raw profile files allow analysis of memory profile data when the memory required for the design plus the memory required for internal profiling data exceeds the memory capacity of the machine.

Loading a new design after you have read the raw profile data clears all internal profile data, but does not turn run-time profiling back on.

Prerequisites

Before using this command, do either of the following:

- Run the [profile on -m -file <filename>](#) command.
- Run the [profile on -m -fileonly <filename>](#) command.

Arguments

- <filename>

(required) Designates the name of the external file in which to save raw profile data.

Related Topics

[profile clear](#)

[profile configure](#)

[profile interval](#)

[profile off](#)

[profile on](#)

[profile open](#)

[profile option](#)

[profile report](#)

[profile save](#)

profile summary

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

profile report

Outputs profiling data that have been gathered up to the point that you execute the 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]  
 [-available] | [-configure]
```

Description

Before using this command, you must enable profiling with the [profile on](#) command.

Arguments

- **-available**
(optional) Reports what is loaded and/or collected.
- **-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/.vhf-filename>:<line#>
- **-configure**
(optional) Generates a report of the current profile configuration.
- **-cutoff <percentage>**
(optional) Filters out entries in the report that had less than <percentage> of time spent in them. The default is to report all entries (for example, 0%) when -cutoff is not specified.
 <percentage> — (required) Any non-negative integer 0 - 100.

- **-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 to save report data 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/.vhf-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, and if the memory profiler was previously enabled and collected any memory information during a run, the profile report will include memory allocation data.
- **-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, and if the performance profiler was previously enabled and profile samples were collected during a run, the profile report will include performance statistics.

- **-ranked**
(optional) Reports a ranked list of statistical performance and memory allocation data.
- **-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 configure](#)
[profile interval](#)
[profile off](#)
[profile on](#)
[profile open](#)
[profile option](#)
[profile reload](#)
[profile save](#)
[profile summary](#)

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

profile save

Saves the profile data to an external database.

Syntax

```
profile save {[-onexit] <filename> | -onexitstatus}
```

Description

The profile save command enables 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. This command is useful for sending profile information to another user for analysis.

For statistical performance profile data, the external database contains the merged data from any simulation runs and profile open commands.

Use the [profile open](#) command to gain access to the database information.

Prerequisites

Use the [profile on](#) command to begin profiling. Profiling must be active before you run profile save, except when you use the -onexit argument.

Arguments

- <filename>
(required) An absolute or relative path and filename to which to save the profile database.
- -onexit
(optional) Causes the command to execute when the simulator exits. You can use this option to queue multiple profile save commands.
- -onexitstatus
(optional) Reports queued profile save -onexit <filename> commands.

Examples

The following sequence of profile save commands shows the difference between immediate profile save commands and those queued with 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 configure](#)
[profile interval](#)
[profile off](#)
[profile on](#)
[profile open](#)
[profile option](#)
[profile reload](#)
[profile report](#)
[profile summary](#)

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

profile status

Returns the status, on or off, for a single specified option. Ignores any additional options after issuing a warning.

Syntax

```
profile status [-p | -m | -assertions | -classes | -cvg | -qdas | -solver]
```

Arguments

- **-p**
(default, if no other option is specified) Reports the status (on or off) of statistical performance profiling.
- **-m**
Reports the status (on or off) of memory allocation profiling.
- **-assertions**
Reports the status (on or off) of fine-grain analysis of memory capacity data collection for assertions and cover directives.
- **-classes**
Reports the status (on or off) of memory capacity data collection for class objects.
- **-cvg**
Reports the status (on or off) of fine-grain analysis of memory capacity data collection for covergroups.
- **-qdas**
Reports the status (on or off) of fine-grain analysis of memory capacity data collection for queues, dynamic arrays, and associative arrays.
- **-solver**
Reports the status (on or off) of fine-grain analysis of memory capacity data collection for randomize () calls.

profile summary

Outputs profiling data into separate categories based on the context in which each callstack is found.

Syntax

```
profile summary [-file <filename>] [-header] [-noheader] [-tcl]
```

Description

The profile summary command accounts for each callstack entry by incrementing the count in the applicable category:

- VHDL — Samples in VHDL files.
- SV — Samples in Verilog files.
- Shared Objects and User C code — PLI code.
- Calls to OS Standard Libraries — shared system libraries like clib, part of the operating system. This includes functions like strcmp (string compare), memcpy (memory copy), and so on.
- Solver — Samples from the solver.
- Assertions — Samples from assertion code.
- WLF file logging — Samples associated with logging data.
- Kernel — Samples that do not fall into any other category.
- Other — Samples in VHDL/SV source code that cannot be mapped to a specific source file.

Each callstack is inspected, starting at the leaf and working up toward the root. The tests for applicable categories occur in the order shown above. The first matching category is incremented for that callstack.

A callstack entry can apply to more than one category, but it is accumulated only into the first matching category. This is why there are VHDL and SV categories separate from the 'other' category. The VHDL/SV categories are used if the first match is from code that can be literally traced to a line in an HDL file. The 'other' category is used for HDL code that does not 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 you specify with profile summary.

Table 3-2. 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

Prerequisites

Before you can use this command you must do one of the following:

- Run the [profile on](#) command.
- Run the vsim -autoprofile command.

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 3-2](#) for more information.
- **-noheader**
(optional) Prevents header data from being written with the summary information. Refer to [Table 3-2](#) for more information.
- **-tcl**
(optional) Output returned as Tcl list.

Examples

profile summary

```
# Profile Summary:
#           kernel    154   43.26%
#           user_c    117   32.87%
#       SystemVerilog    55   15.45%
#           other     30    8.43%
# =====
#           Total    356  100.00%
```

Related Topics

[profile clear](#)

[profile configure](#)

[profile interval](#)

[profile off](#)

[profile on](#)

[profile open](#)

[profile option](#)

[profile reload](#)

[profile report](#)

[profile save](#)

[Profiling Performance and Memory Use \[Questa SIM User's Manual\]](#)

project

Used to perform common operations on projects.

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>]
```

Description

Some arguments to this command are applicable only if you have opened a project with either the project new or project open command. Some arguments are applicable only outside of a simulation session. Refer to the argument descriptions for more information.

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 you do not specify a folder name, the file is 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 process repeats 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 defaults to "work" unless otherwise specified. You cannot create a new project while a project is currently open, or while 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 is saved as an .mpf file

<defaultlibrary> — Specifies a name for the default library.

<intialini> — Specifies an optional *modelsim.ini* file as a seed for the project file. If initialini is an empty string, 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 the 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 change 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

Changes one or more properties of the specified signal, net, or register in the List window.

Syntax

```
property list [-window <wname>] [-label <label>] [-radix <type>]
               [-radixenumnumeric | -radixenumsymbolic] [-trigger 0 | 1] [-width <number>] <pattern>
```

Description

The properties correspond to those you can set by selecting **View > Signal Properties** (List window). You must use at least one argument.

Arguments

- **-window <wname>**
(optional) Specifies a particular List window when multiple instances of the window exist (for example, list2). If you do not specify a window, the default window is used; the most recent invocation of the [view](#) command determines the default window, which 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 you do not specify a radix 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. Refer to [DefaultRadix](#) in the User’s Manual for more information.

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 the final argument to the property list command.

property wave

Changes one or more properties of the specified signal, net, or register in the Wave window.

Syntax

```
property wave [-window <wname>] [-color <color>] [-format {analog | literal | logic}]  
[-height <number>] [-offset <number>] [-radix <type>] [-radixenumsymbolic]  
[-scale <n>] <pattern>
```

Description

The properties correspond to those you can set by selecting **View > Signal Properties** (Wave window). You must use at least one argument.

Arguments

- **-window <wname>**
(optional) Specifies a particular Wave window, when multiple instances of the window exist. If you do not specify a window, the default window is used; the most recent invocation of the [view](#) command determines the default window, which 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 to be used for the waveform.

 <color> — The color value can 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 you do not specify a radix for an enumerated type, the default representation is used.

You can use the [radix](#) command to change the default radix for the current simulation. You can change the default radix permanently by editing the DefaultRadix variable in the *modelsim.ini* file. Refer to [DefaultRadix](#) in the User's Manual for more information.

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 the final argument to the property wave command

push

Moves the specified number of call frames down the C callstack.

Syntax

push <#_of_levels>

Description

This command is used with C Debug. Refer to “[C Debug](#)” in the User’s Manual for more information.

Arguments

- **<#_of_levels>**
(optional) Specifies the number of call frames to move down the C callstack. Defaults to 1 level, if unspecified.

Examples

- Move down 1 call frame.

push

- Move down 4 call frames.

push 4

Related Topics

[pop](#)

[C Debug \[Questa SIM User's Manual\]](#)

pwd

A Tcl command; displays the current directory path in the Transcript window.

Syntax

`pwd`

Arguments

None

questasim

Starts the QuestaSim GUI without prompting you to load a design.

Syntax

questasim [-do <macrofile>] [<license_option>]

Description

This command is valid only on Windows platforms. You can invoke it from:

- the DOS prompt
- a QuestaSim shortcut
- the Windows Start > Run menu

To use questasim arguments with a shortcut, add them to the target line of the shortcut properties. You can also use arguments on the DOS command line.

You can invoke the simulator from either the QuestaSim> prompt, after the GUI starts, or from a DO file called by questasim.

Arguments

- **-do <macrofile>**
(optional) Executes a specified DO file path when questasim is invoked.
 <macrofile> — Any valid DO file path.
- **<license_option>**
(optional) Refer to the vsim [<license_option>] argument.

Note

 In addition to the macro called by this argument, any DO file specified by the STARTUP variable in *modelsim.ini* is 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

[Using a Startup File \[Questa SIM User's Manual\]](#)

quietly

Turns off transcript echoing for the specified command.

Syntax

`quietly <command>`

Arguments

- `<command>`

(required) Specifies the command to disable transcript echoing for. Results that are normally echoed will no longer appear in the Transcript window. To disable echoing for all commands use the [transcript](#) command with the -quietly option.

Related Topics

[transcript](#)

quit

Exits the simulator.

Note

 Do not use either the quit command or an [exit](#) command to stop the simulation using a [when](#) command. Instead, you must use a [stop](#) command in your when statement. 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. Closes all files opened by the simulation, including the WLF file (*vsim.wlf*).
- **-code <integer>**
(optional) Quits the simulation and issues an exit code.

<integer> — The value of the exit code. Do not specify an exit code that already exists in Questa SIM. Refer to "[Exit Codes](#)" in the User's Manual for a list of existing exit codes. You can also specify a variable in place of *<integer>*.

Best practice is to 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 command -code arguments behave the same way.

qverilog

Compiles (vlog), optimizes (vopt), and simulates (vsim), Verilog, and SystemVerilog designs in a single step.

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>]
```

Description

The qverilog command invokes vlog, vopt, and then vsim. It supports all standard vlog (and vopt) arguments, and applies them directly to the qverilog command line. It supports all vsim options and applies them through the qverilog -R argument.

All compiled but unreferenced modules become top level instances. Unreferenced modules compiled in files included by vlog -y also become top level modules, unless you specify -svext=-uslt with the qverilog command.

You can directly enter either C or C++ files onto the qverilog command line. Questa SIM automatically processes them using the SystemVerilog Direct Programming Interface (DPI). Refer to “[DPI and the vlog Command](#)” in the User’s Manual 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, then invokes an implicit "run -all; quit -f" to quit automatically. However, if you invoke qverilog with -do, -gui, or -i, the simulator stays open until you explicitly quit Questa SIM.

If one does not already exist, the qverilog command creates a work library named work in the current directory.

Arguments

- [<vlog_and_vopt_options>]
Supports all vlog and vopt options. For example, to run qverilog on a SystemVerilog design, add the -sv argument to the command line. See the vlog and vopt pages for descriptions of their options.
- -ccflags "opts"
(optional) Specifies all C/C++ compiler options. Place options in quotes.
For -ccflags and -ldflags, qverilog does not check the validity of the option(s) you specify. The options are passed directly to the compiler and linker, and if they are not valid, the compiler/linker generates an error message.
- -ccwarn [on | off | verbose | strict]
(optional) 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 to run the simulator b in interactive mode.
- **-l <logfile>**
(optional) Creates a logfile/transcript compatible with vlog -l logfile. If omitted, a default transcript named *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. Requires at least one filename, and you can enter multiple filenames in a space-separated list. Allows wildcards. In the case of C files, they are automatically processed as DPI code.
- **-ldflags "opts"**
(optional) Specifies all linker options in quotes.

For -ccflags and -ldflags, qverilog does not check the validity of the option(s) you specify. The options are passed directly to the compiler and linker, and if they are not valid, the compiler/linker generates an error message.
- **-R <vsim_options>**
(optional) Specifies valid vsim arguments to apply to the simulation. You must enter all vlog and vopt arguments before -R, 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).

You can specify multiple features and modes for each instance of -stats, 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. The plus character (+) enables the feature, and the minus character (-) disables the feature. You can also specify a feature or mode without the plus (+) character to enable it. Setting this switch adds or subtracts features and modes from the default settings "cmd,msg,time".

Features

all — Display all statistics features (cmd, msg, perf, time). Applied first, when specified in a string with other options. Mutually exclusive with the none option.

cmd — (default) Echo the command line.

msg — (default) Display error and warning summary at the end of command execution.

none — Disable all statistics features. Applied first, when specified in a string with other options. Mutually exclusive with the all option.

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 set globally for all features. Use the plus (+) or minus (-) character to add or subtract a mode for a specific feature; for example, vcover dump -stats=cmd+verbose,perf+list. Use a comma-separated list to add or subtract a mode globally for all features; 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

 You can use -stats=perf+verbose to display information about the operating system and host machine on which the executable is run. You can use qverilog -quiet to disable all default or user-specified -stats features.

- -work <library_name>

(optional) Specifies the logical name or pathname of a library to map to the logical library work. By default, 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 -I 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"
```

- Ignore the first -stats option, disable all *modelsim.ini* settings with the none option, and then enable the perf option.

```
qverilog -stats=time,cmd,msg -stats=none,perf
```

radix

Specifies the default radix to use for the current simulation.

Syntax

```
radix [-binary | -octal | -decimal | -hexadecimal | -HEXADECIMAL | -unsigned | -ascii | -time]
      [-enumnumeric | -enumsymbolic | -showbase | -symbolic | -wreal]
```

Description

You can use the radix command at any time. Specify radix with no argument to return the current radix setting.

The specified radix is used for all commands (such as [force](#), [examine](#), [change](#)), and for values displayed in the Objects, Locals, Dataflow, Schematic, List, and Wave windows, as well as the Source window in source annotation view.

There are two alternate methods for changing the default radix:

- Edit the DefaultRadix variable in the *modelsim.ini* file. (Refer to [DefaultRadix](#) in the User's Manual for more information.)
- Choose **Simulate > Runtime Options** from the main menu, click the **Defaults** tab, make your selection in the **Default Radix** box.

Numeric radix, other than symbolic, 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 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.

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 display 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. Reverses 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 using lower-case a-f characters.
- **-HEXADECIMAL**
(optional) Case-sensitive argument that displays values using upper-case A-F characters.
- **-octal**
(optional) Displays values in octal format.
- **-time**
(optional) Displays values of time for register-based types in Verilog.
- **-showbase**
(optional) Displays the number of bits of the vector and the radix used (binary = b, decimal = d, hexadecimal = h, ASCII = a, and time = t)

For example, a vector value of “31” might display as “16’h31” to show that the vector is 16 bits wide, with a hexadecimal radix.
- **-symbolic**
(optional) Displays values in a form closest to their natural form.
- **-unsigned**
(optional) Displays values in unsigned decimal format.
- **-wreal**
(optional) Displays internal values of real numbers that are determined to be not-a-number (NaN) as X or Z instead of as "nan." Internal values of NaN that match `wrealZState, display a 'Z'; otherwise, an 'X' is displayed.

Related Topics

[User-Defined Radices \[Questa SIM GUI Reference Manual\]](#)

[radix define](#)

[radix delete](#)

[radix names](#)

[radix list](#)

[radix signal](#)

radix define

Creates or modifies a user-defined radix.

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>  
}
```

- <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.
```

You can use the question mark (?) wildcard character 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 matches "010000", "010100", "011000", and "011100". In case of overlaps, the first matching pattern is used, going from top to bottom.

- <enum-label> Any arbitrary string. Enclose it in quotation marks (""), especially if it contains spaces.
- The comma (,) in the definition body is optional.

- -color <color> Sets the color to use to display the specific value in the Wave window.
- -default <radix_type> (optional) 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.
- -defaultcolor <color> (optional) Defines the default color to use if none has been defined for a specific value.

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 can be a color name or its hex value (see example in the Example section, 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, specifying -fraction 3 for the eight bit vector “10001001” places the decimal 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 the number of places after the decimal point or significant digits of a floating point or fixed number in symbolic format.

 <p> — A number, less than or equal to 17, and an 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.

Description

You can use a user definable radix to map bit patterns to a set of enumeration labels or to set up a fixed or floating point radix. User-defined radices are available for use in most windows, and with the examine command.

Examples

- 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'b000000000001 "IDLE",
    11'b000000000010 "CTRL",
    11'b000000000100 "WT_WD_1",
    11'b000000001000 "WT_WD_2",
    11'b000000010000 "WT_BLK_1",
    11'b000000100000 "WT_BLK_2",
    11'b000010000000 "WT_BLK_3",
    11'b000100000000 "WT_BLK_4",
    11'b001000000000 "WT_BLK_5",
    11'b010000000000 "RD_WD_1",
    11'b100000000000 "RD_WD_2",
    11'bzzzzzzzzzzzz "UNCONNECTED",
    11'bxxxxxxxxxxx "ERROR",
    -default hex
}
```

Note

 The ‘z’ and ‘x’ values must be lower case.

- Specify the radix color:

```
radix define States {
    11'b000000000001 "IDLE" -color yellow,
    11'b000000000010 "CTRL" -color #ffee00,
    11'b000000000100 "WT_WD_1" -color orange,
    11'b000000001000 "WT_WD_2" -color orange,
    11'b000000010000 "WT_BLK_1",
    11'b000000100000 "WT_BLK_2",
    11'b000010000000 "WT_BLK_3",
    11'b000100000000 "WT_BLK_4",
    11'b001000000000 "WT_BLK_5",
    11'b010000000000 "RD_WD_1" -color green,
    11'b100000000000 "RD_WD_2" -color green,
    -default hex
    -defaultcolor white
}
```

If a pattern/label pair does not specify a color, the normal wave window colors are used. If the value of the waveform does not match any pattern, the -default radix and -defaultcolor are 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 \[Questa SIM GUI Reference Manual\]](#)

[precision](#)

[radix](#)

[radix delete](#)

[radix names](#)

[radix list](#)

[radix signal](#)

radix delete

Removes 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 \[Questa SIM GUI Reference Manual\]](#)

[radix](#)

[radix define](#)

[radix list](#)

[radix names](#)

[radix signal](#)

radix list

Returns the complete definition of a radix, if a name is given. Lists all defined radices, if no name is given.

Syntax

`radix list [<name>]`

Arguments

- `<name>`
(optional) Returns the complete definition of the named radix.

Related Topics

[User-Defined Radices \[Questa SIM GUI Reference Manual\]](#)

[radix](#)

[radix define](#)

[radix delete](#)

[radix names](#)

[radix signal](#)

radix names

Returns a list of currently defined radix names.

Syntax

`radix names`

Arguments

None

Related Topics

[User-Defined Radices \[Questa SIM GUI Reference Manual\]](#)

[radix](#)

[radix define](#)

[radix delete](#)

[radix list](#)

[radix signal](#)

radix signal

Sets or inspects radix values for the specified signal in the Objects, Locals, Schematic, and Wave windows, or returns a list of all signals with a radix, when entered with no argument.

Note

 This command is most useful for a small number of signals. If the majority of signals in a design are to use a particular radix value, use the [radix](#) command to set that value as the default radix, and use the radix signal command for the rest.

Syntax

`radix signal [<signal_name> [<radix_value>]] [-showbase]`

Arguments

- `<signal_name>`
(optional) Name of the signal for which to set the radix (if `<radix_value>` is specified) or inspected.
- `<radix_value>`
(optional) Value of the radix to set for the specified signal. Use empty quotation marks ("") to unset the radix for the specified signal.
- `-showbase`
(optional) Display the number of bits of the vector, and the radix used (binary = b, decimal = d, hexadecimal = h, ASCII = a, and time = t).

For example, instead of displaying a vector value of “31”, display a value of “16'h31” to show that the vector is 16 bits wide, with a hexadecimal radix.

Related Topics

[User-Defined Radices \[Questa SIM GUI Reference Manual\]](#)

[radix](#)

[radix define](#)

[radix list](#)

[radix delete](#)

readers

Displays the names of all readers of the specified object.

Syntax

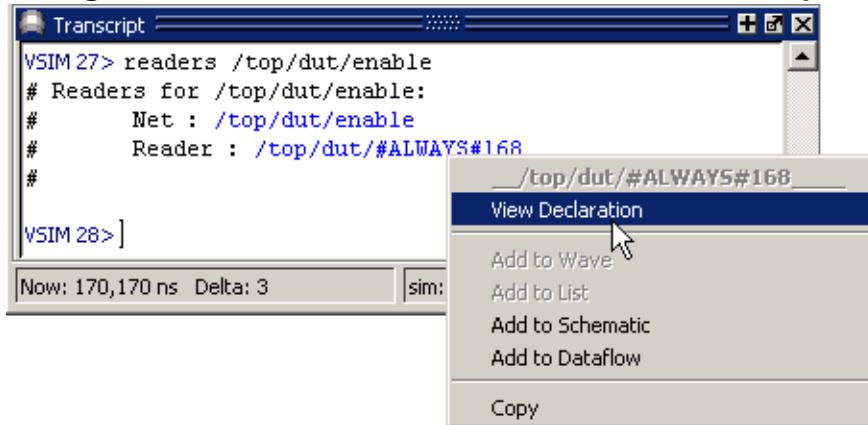
readers <object_name> [-source]

Description

The reader list is expressed relative to the top-most design signal/net connected to the specified object.

The output from the readers command displays in the Transcript window as hypertext links. You can right-click a link to open a drop-down menu and add signals to various windows. The drop-down menu includes a "View Declaration" item to open the source definition of the signal.

Figure 3-1. readers Command Results in Transcript



Arguments

- <object_name>
(required) Specifies the name of the signal or net to show readers for. All signal or net types are valid. Accepts multiple names and wildcards.
- -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.

report

Displays information relevant to the current simulation.

Note

 The report command does not display information on preference variables. To view preference variables, select **Tools > Edit Preferences** (Main window) to open the Preferences dialog box.

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, with the arguments limiting the list to the specified files. 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 with the Tcl [set Command Syntax](#), described in the User's Manual.

- Display all simulator state variables. (Displays only the variables that relate to the design being simulated):

report simulator state

Returns:

```
#now = 0.0
#delta = 0
#library = work
#entity = type_clocks
#architecture = full
#resolution = 1ns
```

Related Topics

[modelsim.ini Variables \[Questa SIM User's Manual\]](#)

“Setting GUI Preferences” [Questa SIM GUI Reference Manual]

restart

Reloads the design elements and resets the simulation time to zero.

Syntax

```
restart [-force] [-nobreakpoint] [-nolist] [-nolog] [-nowave] [-noassertions] [-nofcovers] [-  
sv_seed <seed_value> | random]
```

Description

Invoking the restart command produces the following behavior:

- If no design is loaded, the command produces a message to that effect and takes no further action.
- If a simulation is loaded, the command restarts the simulation.
- If multiple datasets are open, including a simulation, the environment changes to the simulation context, and the simulation restarts.

The restart command reloads only design elements that have changed. (SDF files are always reread during a restart.)

Shared libraries are handled as follows during a restart:

- Shared libraries that implement only VHDL foreign architectures 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 set the `DefaultRestartOptions` variable in the `modelsim.ini` file to configure defaults for the command. Refer to “[Restart Command Defaults](#)” in the User’s Manual.

To handle restarts with Verilog PLI applications, you must define a Verilog user-defined task or function, and register a `misctf` class of callback.

To handle restarts with Verilog VPI applications, you must register reset callbacks. To handle restarts with VHDL FLI applications, you must register restart callbacks.

Refer to “[Verilog Interfaces to C](#)” in the User’s Manual for more information on the Verilog HDL interfaces.

Refer to the *Foreign Language Interface Manual* for more information on the FLI.

Arguments

- **-force**
(optional) Specifies to restart the simulation without requiring confirmation in a popup window.
- **-noassertions**
(optional) Specifies not to maintain the current assert directive configurations after restarting the simulation. The default is to maintain assert directive settings.
- **-nobreakpoint**
(optional) Specifies to remove all breakpoints when restarting the simulation. The default is to reinstall all breakpoints.
- **-nofcovers**
(optional) Specifies not to maintain current cover directive configurations after restarting the simulation. The default is to maintain cover directive settings.
- **-nolist**
(optional) Specifies not to maintain the current List window environment after restarting the simulation. The default is to maintain all currently listed HDL objects and their formats.
- **-nolog**
(optional) Specifies not to maintain the current logging environment after restarting the simulation. The default is to continue logging all currently logged objects.
- **-nowave**
(optional) Specifies not to maintain the current Wave window environment after restarting the simulation. The default is for all objects displayed in the Wave window to remain in the window with the same format.
- **-sv_seed <seed_value> | random**
(optional) Uses either the real number you specify for <seed_value> or a random number (selected by Questa SIM) as the seed for the random number generator.

Related Topics

[restore](#)

restore

Restores the state of a simulation that was saved with a checkpoint command during the current invocation of vsim (a “warm restore”).

Syntax

restore <pathname>

Description

The restored items 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.

Use this command to restore while running vsim. To start up vsim and simultaneously 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. You cannot warm restore checkpoint files that were not created in the current run, except for this special case of an original cold restore file.

Restrictions for this command include:

- Checkpoint files are platform dependent—you cannot run checkpoint on one platform and restore on another.
- Most designs containing SystemC do not support the restore command once the design is loaded. However, some SystemC modules do support the use of the restore command with vsim -scchkpntrestore. For more information, refer to [Checkpoint and Restore in a SystemC Design](#) in the User’s Manual.

Arguments

- <pathname>
(required) Specifies the pathname of the checkpoint directory or the checkpoint file previously saved with the ‘checkpoint’ command.

Related Topics

[Checkpointing and Restoring Simulations \[Questa SIM User's Manual\]](#)

resume

Resumes execution of a macro (DO) file after a pause command or a breakpoint.

Syntax

`resume`

Description

You can enter this command manually, or place it in an [onbreak](#) command string. (Placing a resume command in a [bp](#) command string does not have the desired effect.) You can also use the resume command in an [onerror](#) command string, to allow an error message to print without halting the execution of the macro file.

Arguments

None

Related Topics

[pause](#)

[Useful Commands for Handling Breakpoints and Errors \[Questa SIM User's Manual\]](#)

right

Searches right (next) for signal transitions or values in the specified Wave window.

Note

 “Wave Window Mouse and Keyboard Shortcuts,” described in the User’s Manual, 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>]
```

Description

The right command 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.

This command enables you to move to consecutive transitions, find the time at which a waveform takes on a particular value, or to find the time at which an expression of multiple signals evaluates to true. See the [left](#) command for related functionality.

To use the right command:

1. Click the desired waveform.
2. Click 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>

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 can involve more than one signal, but is limited to signals that have been logged in the referenced Wave window. You can specify a signal 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. You must enclose the expression in curly braces.

- **-falling**

(optional) Searches for a falling edge on the specified scalar signal. Ignores non-scalar signals.

- **-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 scalar signal. Ignores non-scalar signals.
- **-value <sig_value>**
(optional) Specifies a value of the signal to match. You can select only one signal, but the signal can be an array.
 <sig_value> — Must be specified in the same radix as that used to display the selected waveform. 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. If not specified, 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* has 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* equal 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 has 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

[seetime](#)

run

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]}
```

Description

You can use the following preference variables to control any return values after the run operation completes:

- 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 information, or set it to 1 to disable the display of this information.

The following example 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>
```

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 with the RunLength and UserTimeUnit *modelsim.ini* file variables. (Refer to [RunLength](#) and [UserTimeUnit](#) in the User's Manual.)

- <timesteps>[<time_units>]

(optional) Specifies the number of timesteps for the simulation to run. The number can be fractional, or can be specified as absolute by preceding the value with the character @.

<time_units> — Any valid time unit: fs, ps, ns, us, ms, or sec. 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 can be input manually, or can be 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. (Refer to [C Debug](#) in the User's Manual.)

- -init

(optional) Initializes non-trivial static SystemVerilog variables before beginning the simulation; for example, expressions involving other variables and function calls. This can be useful 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](#) -rununit on the command line, because all variables 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.

You can observe current values of local HDL variables in the Locals window at this time. 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 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.

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 \[Questa SIM GUI Reference Manual\]](#)

[step](#)

runStatus

Returns the current state of your simulation to stdout after a run or step command.

Syntax

`runStatus [-full]`

Arguments

- `-full`

(optional) Appends additional information to the output of the runStatus command.

Return Values

[Table 3-3 \(runStatus Command States\)](#) and [Table 3-4 \(runStatus -full Command Information\)](#) show outputs of the runStatus command.

Table 3-3. 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 3-4. 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 3-4. 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 or equivalent.
gate_oscillation	Verilog gate iteration limit reached.
simulation_stop	pli stop_simulation() called.

sccom

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.

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] [-sctop <sc_module_name>] [-scv] [-  
scversion] [-stats [=+ | -]<arg>[,[+ | -]<arg>]] [-suppress {<msgNumber> | <msgGroup>}  
  [,<msg_number> | <msgGroup>,...]] [-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] [-uvmc] [-vv] [-verbose] [-  
version] [-work <library_name>]
```

Description

You can distribute the compilation of multiple SystemC source files across multiple machines, or use the *j* argument to distribute compilation across multiple cores on a single machine. The result is faster compilation.

You can run this command in Questa SIM, or 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 must run vdel -allsystemc on the working library and then recompile your SystemC source.
- Version — If you install a new release, you must 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 recognizes the file type as either C or C++ by the filename extension and uses 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, run sccom in multi-file compilation mode, which requires that you 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.

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 specify these options only on the command line; 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>

You can use any normal C++ compiler option, 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 edit the CppOptions variable in the *modelsim.ini* file to specify arguments for all sccom compiles.

- -DSC_

(optional) Specifies SystemVerilog libraries for SystemC DPI (Direct Programming Interface). You can specify only one library for each -dpilib argument. Refer to

[SystemC Procedural Interface to SystemVerilog](#) in the User's Manual for more information.

- -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>

You can use any normal C++ compiler option, with the exception of the -o option. You can edit the CppOptions variable in the *modelsim.ini* file to specify arguments for all sccom compiles.

- -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
```

When the -cpppath argument is also present, the order of precedence in determining the compiler path is the following:

- The -cppinstall argument is used with sccom
- The -cpppath argument is used with sccom
- -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.

When the -cppinstall argument is also present, the order of precedence in determining the compiler path is the following:

- The -cppinstall argument is used with vopt
- The -cpppath argument is used with vopt
- -dumpsCsvExt <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). You can specify only one library for each -dpilib argument. Refer to [SystemC Procedural Interface to SystemVerilog](#) in the User's Manual for more information.
- -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](#)" in the User's Manual for more information.
- [-F | -file | -f] <filename>
(optional) -f, -file and -F: each specifies an argument file with more command-line arguments, allowing reuse of complex argument strings without retyping. Nesting of -F, -f and -file commands is allowed. Allows gzipped input files.
With -F only: when lookup with relative path fails, relative file names and paths within the arguments file <filename> are prefixed with the path of the arguments file. Refer to the section "[Argument Files](#)" on page 37" 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](#)" in the User's Manual 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 "[Verifying Compiler Information](#)" in the User's Manual for more information.
- -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 differs 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 re-compilation of a file when the only changes are to access or modification time (touch), or to comments (except that changes to source code that affect line numbers, such as adding a comment line, do cause recompilation of all affected files). This keeps 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 you use a different version of gcc 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>**

(optional) Specifies the default working library where the SystemC linker can find the object files for compiled SystemC modules. Use only with sccom -link.

- **-libshared <library_pathname>**

(UNIX, Linux only; optional) Specifies the library location for the intermediate SystemC library generated with sccom -linkshared.

- **-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. There are two types of dependencies, and placement of the -link argument is 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 -link
```

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> libsystmc.a a.o b.o libtemp.a
```

- **-linkshared**
(UNIX, Linux only; optional) Creates an intermediate SystemC shared library, which allows you to create multiple SystemC shared libraries (systemc.so). The simulator cannot elaborate individual intermediate shared objects. All intermediate shared objects generated using sccom -linkshared need to be linked into the final systemc.so during the sccom –link step using the -libshared <lib> option. You cannot use this argument 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 filename. 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 SystemC modules. Using -nodebug can significantly reduce the compile time for a design, which is useful when running designs in regression mode. Specify -nodebug at both sccom compile and at link time. If any source file contains the SC_MODULE_EXPORT() macro, you must use -nodebug in conjunction

with the -sctop argument to prevent mixed-language designs from compiling incorrectly.

Limitation: Using -nodebug in a VHDL instantiating SystemC scenario causes vsim to return an error. 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](#)" in the User's Manual for more information.
- **-optionset <optionset_name>**
(optional) Calls an optionset as defined in the *modelsim.ini* file. Refer to [Optionsets](#) 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 "predef_macro_gcc<COMPILER_VERSION>_<PLATFORM>.txt" file, which you must then copy over to the location *\$MTI_HOME/include/systemc*. Use of -predefmacrofile makes it unnecessary to copy 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 can also enable SystemC-2.2 with the Sc22Mode *modelsim.ini* variable. Refer to [Sc22Mode](#) in the User's Manual.
- **-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.
- **-sctop <sc_module_name>**
(optional) Use this argument with the -nodebug argument to specify the SystemC top level design unit(s). Every SystemC module you specify with the SC_EXPORT_MODULE(<sc_module_name>) macro, you must also specify on the sccom command line with the -sctop <sc_module_name> argument during compilation. For example: sccom *.cpp -nodebug -sctop <sc_module_name_1> -sctop <sc_module_name_2> ... <other sccom options>. Use this argument only in conjunction with -nodebug.

- **-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).

Use a comma-separated list to specify multiple features and modes for each instance of -stats. 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. The plus character (+) enables the feature, and the minus character (-) disables the feature. You can also enable a feature or mode by specifying it without the plus (+) character. Setting this switch adds or subtracts 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

 Use -stats=perf+verbose to display information about the operating system and host machine.

Note

 You can specify -quiet to disable display of all statistics during compile.

- **-suppress {<msgNumber> | <msgGroup>} [,,<msg_number> | <msgGroup> ,...]**
 (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 [suppress](#) in the User's Manual.

<msgNumber>

A specific message number

<msgGroup>

A value identifying a pre-defined group of messages, based on area of functionality.
 These groups only suppress notes or warnings. The valid arguments are:

All, GroupNote, GroupWarning, GroupFLI, GroupPLI, GroupSDF, GroupVCD,
 GroupVital, GroupWLF, GroupTCHK, GroupPA, GroupLRM

- **-uvmc**

(optional) Enables automatic linking of the precompiled UVMC libraries that ship with Questa SIM. This argument is a link-time only option so you can use it only with sccom -link. You can also enable this linking by specifying the UseUvmc variable in the *modelsim.ini* file.

Note



The -uvmc argument for sccom is not supported for Windows.

- **-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. Refer to [warning](#) in the User's Manual.

- **-work <library_name>**

For the compiler — Specifies a logical name or pathname of a library to map 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 in which to store the final linked object file (.so). 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 you are compiling, when the filename extension does not match that source. For example, if myfile.cpp contains 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 the file containing the SystemC/C++ source to compile. You can enter multiple filenames, separated by spaces. You can also use wildcards to specify multiple filenames (such as *.cpp).

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*.)

```
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, use the -sclib switch with the **vsim** command to specify the .so library. The -lib switch tells the simulator where the top-level module is compiled. The 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 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. Refer to “[Differences Between the Simulator and OSCI](#)” in the User’s Manual for more information.

sccom -I/home/systemc/include -DSC_INCLUDE_FX -g a.cpp b.cpp

- Compile with the g++ -O2 optimization argument.

sccom -O2 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 \[Questa SIM User’s Manual\]](#)

[scgenmod](#)

scgenmod

Takes a Verilog or VHDL module that has been compiled into a library, and writes its equivalent SystemC foreign module declaration to standard output.

Note

 An 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>"] [-mapvectoint] [-modelsimini <ini_filepath>] [-createtemplate] [-preservevhdlcase] [<scalar_type>] [<vector_type>] <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. Enables ports in VHDL and Verilog modules instantiated from SystemC to have their range specified in terms of integer parameters/generics. Such port ranges are 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.
- **-mapvectoint**
Maps HDL vectors of appropriate sizes to C++ integer types.
- **-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_bignum, 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 filename. 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.
- **<scalar_type>**
Specifies the type of scalar port to be generated.
 - sc_logic**
Generate sc_logic scalar port types
 - sc_bit**
Generate sc_bit scalar port types.
 - bool**
Generate bool scalar port types.
 - sc_resolved**
Generate resolved scalar port types.
- **<vector_type>**
Specifies the type of vector port to be generated.
 - sc_lv**

Generate sc_lv<N> vector port types.

-sc_bv

Generate sc_bv<N> vector port types.

-sc_rv

Generate resolved vector port types.

-sc_int

Generate sc_int<N> vector port types.

-sc_uint

Generate sc_uint<N> vector port types.

- <module_name>

(required) Specifies the name of the Verilog/VHDL module to access.

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_name);}
    ~vcounter()
    {}
};
```

Related Topics

[SystemC Simulation \[Questa SIM User's Manual\]](#)

[sccom](#)

sdfcom

Compiles the specified SDF file.

Syntax

```
sdfcom [-delayscale <value>] [-maxdelays] [-mindelays] [-nocompress] [-suppress <msg_list>]  
[-typdelays] <source_file> <output_file>
```

Description

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). 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.

Arguments

- [-32 | -64]

(optional) Specifies whether the command 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 this command from an <install_dir>/<platform>/ directory.

You can only specify these options on the command line, and are not recognized as part of a file used with the -f switch.

- -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](#), or 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).

- **-suppress <msg_list>**
Suppress error and warning messages, where <msg_list> is a comma-separated list of message numbers.
- **-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 the final argument to the sdfcom command.

Related Topics

[Compiling SDF Files \[Questa SIM User's Manual\]](#)

search

Searches the specified window for one or more objects matching the specified pattern(s).

Syntax

```
search <window_name> [-window <wname>] [-all] [-field <n>] [-toggle]
    [-forward | -backward] [-wrap | -nowrap] [-exact] [-regexp] [-nocase] [-count <n>]
    <pattern>
```

Description

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.

Note

 You can use the next command to continue the search.

Returns the index of a single match, or a list of matching indexes. Returns nothing if no matches are found.

With the **-all** option, the command searches the entire window, selects the last object matching the pattern, and returns a Tcl list of all corresponding indexes.

With the **-toggle** option, the command selects objects found in addition to the current selection.

For the List window, the search is done on the names of the listed objects, 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.

Arguments

- **<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 the first argument to the search command.
- **-window <wname>**
(optional) Specifies a particular window when multiple instances of the window exist. Uses the default window, if no window is specified; 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 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 3-5. 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, to 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.

searchlog

Sets one or more of the currently open logfiles for a specified condition.

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

You can use this command to search for rising or falling edges, for signals equal to a specified value, or for when a generalized expression becomes true. If the search finds at least one match, the search 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 you specify the -deltas option, the delta of the last match is also returned:

```
{ {<time>} <delta> <matchCount> }
```

If the search does not find any matches, it returns a TCL_ERROR. Finding one or more matches, but less than the requested number, 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 to 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 is passed four arguments: the reason for the call, the time of the event, the delta for the event, and the value. The reason value is 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 returns to the caller of the searchlog command.

The command supports searching for only a single signal.

- **-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. When searching backwards, you must specify this option after **-reverse**.

<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. The default is to specify the current simulation resolution by omitting <unit>. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}). Valid VHDL time units are: fs, ps, ns, us, ms, sec, min, and hr.

For **-reverse** searches, the specified end time must be earlier than the specified <startTme>.

For **-reverse** searches the default <endTme> is 0.

- **-env <path>**

(optional) Specifies a path to a design region. You cannot use wildcards.

- **-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** searches until the expression evaluates to true.

{<expr>} — An expression that evaluates to a boolean true. See “[GUI_expression_format](#)” on page 42 for the format of the expression.

- **-reverse**

(optional) Specifies to search backwards in time from <startTme>. 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. Ignored for compound signals.

- **-falling**

(optional) Specifies a search for falling edge on a scalar signal. Ignored for compound signals.

- **-anyedge**

(optional) Specifies a search for a rising or falling edge on a scalar signal. 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. The default is to specify the current simulation resolution by omitting <unit>. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}). Valid VHDL time units are: fs, ps, ns, us, ms, sec, min, and hr.
- **<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 the final argument to the searchlog command.

see

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, displays five lines 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

```
# 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

Scrolls the List or Wave window to make the specified time visible.

Syntax

```
seetime list | wave [-window <wname>] [-select] [-delta <num>] [-event] <time> [<unit>]]
```

Description

Returns the specified time and unit. You can optionally specify a delta for the List window.

Arguments

- **list | wave**

(required) Controls which type of window to target. Must be the first argument to the seetime command.

list — Target a list window.

wave — Target a wave window.

- **-window <wname>**

(optional) Specifies to use a window other than the current default Wave or List window. The default, when -window is not specified, is to use the current default Wave or List window. 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) Moves the active cursor or marker to the specified time (and optionally, delta). Otherwise, the window is only scrolled.

- **-delta <num>**

(optional) Specifies a delta for the List window, when deltas are not collapsed. Otherwise, delta 0 is selected.

<num> — Any positive integer.

- **-event**

(optional) Scrolls the List or Wave window 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 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>. If <unit> is specified, you must enclose <time> and <unit> within curly braces ({}). Valid VHDL time units are: fs, ps, ns, us, ms, sec, min, and hr.

setenv

Changes or reports the current value of an environment variable.

Note

 The setting is valid only for the current Questa SIM session.

Syntax

`setenv <varname> [<value>]`

Arguments

- `<varname>`
(required) The name of the environment variable to set or check. Must be 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

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.

Syntax

shift

Description

The shift command and macro parameters are used in macro files. When a macro file requires more than nine parameters, you can use the shift command to access them.

To determine the current number of macro parameters, use the argc simulator state variable. Refer to “[Simulator State Variables](#)” in the User’s Manual for more information.

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.

Arguments

None

show

Lists HDL objects and subregions visible from the current environment.

Syntax

show [-all] [<pathname>]

Description

The list of objects includes:

- VHDL — signals, processes, constants, variables, and instances.
- Verilog — nets, registers, tasks, functions, instances, variables, and memories.

When you are 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.

Arguments

- **-all**
(optional) Displays all names at and below the specified path recursively.
- **<pathname>**
(optional) Specifies the pathname of the environment for which to list the objects and subregions; if omitted, the current environment is assumed.

Examples

- List the names of all objects and subregion environments visible in the current environment.
show
- List the names of all objects and subregions visible in the environment named /uut.
show /uut
- List the names of all 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 objects and subregions visible in all top-level environments.
show -all /

simstats

Returns performance-related statistics about elaboration and simulation with the data for each statistic on a separate line.

Syntax

```
simstats [covsavecpu | covsavetime | elabcpu | elabmemory | elabtime | license | logcpu | logtime  
| simcpu | simmemory | simtime | tlcmdcpu | tlcmdtime | totalcpu | totaltime | verbose |  
voptcpu | voptmemory | voptime] [kb]
```

Description

The statistics measure the simulation kernel process 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 on separate lines. For example:

```
# Memory Statistics  
#     mem: size after elab (VSZ)          88.89 Mb  
#     mem: size during sim (VSZ)           97.17 Mb  
# 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
```

You can use the [simstatslist](#) command to provide this output as a continuous display (without line breaks).

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 auto-scaled.

Note

 Different operating systems report these numbers differently.

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.
- **kb**
(optional) Returns statistics in kilobyte units with no auto-scaling.
- **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. Refer to [Two-Step Flow](#) in the User’s Manual.)
 - **voptmemory**
(optional) Returns memory consumed by vopt process. (Applicable in Two-Step Flow.)
 - **voptime**
(optional) Returns wall clock time of vopt process. (Applicable in Two-Step Flow.)

simstatslist

Returns performance-related statistics about elaboration and simulation as a continuous list (without line breaks).

Syntax

```
simstatslist [covsavecpu | covsavetime | elabcpu | elabmemory | elabtime | logcpu | logtime |  
simcpu | simmemory | simtime | tclcmdcpu | tclcmdtime | totalcpu | totaltime | voptcpu |  
voptmemory | vopttime]
```

Description

Use this command in place of the [simstats](#) command to display the original statistics in a continuous format (without line breaks). For example:

```
 {{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 a description of each statistic.

Units for time values are in seconds. Units for memory values are in kilobytes.

Note

 Different operating systems report these numbers differently.

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 Two-Step Flow. Refer to [Two-Step Flow](#) in the User's Manual.)
- **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

Moves down the call stack.

Syntax

`stack down [n]`

Description

If invoked without arguments, the command moves down the call stack by 1 level. The Locals window displays local variables at the level.

Arguments

- `n`

(optional) Moves down the call stack by *n* levels. The default value is 1.

Related Topics

[stack frame](#)

[stack level](#)

[stack tb](#)

[stack up](#)

stack frame

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

Reports the current call frame number.

Syntax

`stack level`

Arguments

None

Related Topics

[stack down](#)

[stack frame](#)

[stack tb](#)

[stack up](#)

stack tb

Displays a stack trace in the Transcript window, listing the sequence of HDL function calls that have been entered to arrive at the current state for the active process.

Note

 The tb command is an alias for the stack tb command.

Syntax

`tb [<#_of_levels>]`

Description

If you are using C Debug, tb displays a stack trace of the C call stack.

Arguments

- `<#_of_levels>`

(optional) Specifies the number of call frames in the C stack to display. Displays the entire C stack, if you do not specify a level. This argument is available only for C Debug.

stack up

Moves up the call stack.

Syntax

`stack up [n]`

Description

If invoked without arguments, the command moves up the call stack by 1 level. The Locals window displays local variables at the level.

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

Lists summary information about currently interrupted macros.

Syntax

status [[file](#) | [line](#)]

Description

If invoked without arguments, the command returns the filename and line number of each interrupted macro, as well as information about which command within the macro was interrupted. It also displays any [onbreak](#) or [onerror](#) commands that have been defined for each interrupted macro.

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

[pause](#)

[resume](#)

step

An alias for the run command with the -step switch. Steps the simulator to the next HDL or C statement, enabling you to observe current values of local HDL variables in the Locals window.

Syntax

```
step [-current] [<n>] [-out] [-over [<n>]] [-this "this==<class_handle>"]
```

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 information or set it to 1 to disable the display of this information.

The following example 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>
```

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. 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.

Related Topics

[run](#)

[Stepping Through Your Design \[Questa SIM User's Manual\]](#)

stop

Used with the when command to stop simulation in batch files.

Syntax

```
stop [-sync]
```

Description

The stop command has the same effect as hitting a breakpoint. You can place the stop command anywhere within the body of the when command.

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 use a [when](#) command to stop the simulation, 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.

Arguments

- -sync

(optional) Stops the currently running simulation at the next time step.

Related Topics

[resume](#)

[run](#)

suppress

Prevents one or more specified messages from displaying.

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.

Syntax

```
suppress [-clear <msg_number>[,<msg_number>,...]] [<msg_number>[,<msg_number>,...]]  
          [<code_string>[,<code_string>,...]]
```

Description

You cannot suppress Fatal or Internal messages. If you enter the suppress command without arguments, it returns the message numbers of all suppressed messages. Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to [suppress](#) and “[Message Severity Level](#)” in the User’s Manual for more information.

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. Use a comma-separated list to specify multiple message numbers.
- `<msg_number>[,<msg_number>,...]`
(optional) A number identifying the message to be suppressed. Use a comma-separated list to specify multiple message numbers.
- `<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

sv_reseed

Reseeds every SystemVerilog random number generator (thread objects, class objects) using either the seed value you specify or a random seed.

Usage

sv_reseed <integer> | random

Description

You can use this command after a vsim -restore (or at any time) to modify the behavior of SystemVerilog random constructs (such as randomize, randcase, randsequence, or \$urandom) beginning from the point the sv_reseed command is issued.

Note



There is a small performance effect (both speed and memory) on a simulation that has been reseeded.

Arguments

- <integer> | random

Specify the seed to be used by the random number generator.

<integer> — Any 32-bit signed integer (from -2147483648 to +2147483647), where the default is 0. Integers outside the value range are assigned either the minimum or maximum value. If you specify an invalid value, a suppressible error (vsim-7031) is issued. If you downgrade this error, the sv_seed value will reflect the 32-bit truncated value of the invalid seed, and that value will be reported in a subsequent vsim-7031 message.

random — A literal string that causes Questa SIM to select a signed integer at random and apply it as the seed for the random number generator.

tb

Displays a stack trace for the current process in the Transcript window.

Syntax

tb [<#_of_levels>]

Description

The tb (traceback) command lists the sequence of HDL function calls that have been entered to arrive at the current state for the active process. The tb command is an alias for the stack tb command.

If you are using C Debug, tb displays a stack trace of the C call stack. Refer to the User's Manual for more [C Debug](#) information.

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

Works in tandem with tcheck_status to report on and enable/disable individual timing checks. Modifies either a check's reporting or X-generation status, and reports the new setting in the Transcript window.

Syntax

```
tcheck_set [-quiet] <instance> [{-m | -n| -p}] [-r [-v]] [<tcheck> | ALL]
[<Stat> | <MsgStat> <XStat>]
```

Description

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 the 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 you can modify those values with tcheck_set.

The tcheck_set command supports the use of asterisks (*) and ellipses (...) as wildcards for defining hierarchy. An asterisk matches any instance in the current scope. An ellipsis matches any instance in the hierarchy below the current scope. When you use wildcards, all timing checks, up to a maximum of six, must be consecutive, and not intermingled with other arguments. If you are providing more than one timing check, you must place all timing checks in parentheses and double quotes, for example; "(SETUP)". Refer to the Examples section, below, for descriptions of wildcard usage.

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 are not Level-1 optimized. See "Examples" below for further information.

Note

 You will receive a warning message if you use tcheck_set or the tcheck_status command on instances compiled with -nodebug.

Arguments

- <instance>
(required) Specifies the instance for which you want to change the reporting or X-generation status.
- -m | -n| -p
(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.
 - p — Indicates that the instance is a hierarchical pathname to a port, and that tcheck_set will operate only on the instance that contains the specified port.
- -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 can 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

- Use wildcards to specify hierarchical path instance names in a hierarchy with four instance levels, where level 1 is Top, below Top is level 2, including Mid1 and Mid2, below Mid1 is level 3, including Bot1 and Bot2, and below Bot1 is level 4, including U1, U2, and U3.

Various ways to specify U1:

```
/Top/Mid1/Bot1/U1
/Top/*/Bot1/U1
/Top/**/U1
/*/*/Bot1/U1
/*/*/*/U1
```

Various ways to specify U, U2, and U3, inclusive:

```
/Top/Mid1/Bot1/*
/Top/Mid1/*
/*/*/Bot1/*
/*/*/*/*
```

Various ways to apply timing check on all instances at level 2 (Mid1 and Mid2):

```
/*/*
/Top/*
```

Various ways to apply timing check on all instances at level 2 and below (Mid1 subtree and Mid2 subtree):

```
/*/...
/Top/...
```

Examples of using the instance names to avoid typing the full path:

```
tcheck_status /*/*/*/U1
tcheck_set /*/*/*/U1 "(SETUP)" "(HOLD)" On
tcheck_set /Top/Mid1/Bot1/U1 "( WIDTH (negedge CLK) )" OFF
tcheck_set /Top/*/*/U1 "( WIDTH (negedge CLK) )" "(SETUP)" OFF
tcheck_set /Top/*/*/... "( WIDTH (negedge CLK) )" "(SETUP)" OFF
```

Related Topics

[tcheck_status](#)

[Compiling and Simulating with Accelerated VITAL Packages \[Questa SIM User's Manual\]](#)

[SDF Timing Annotation \[Questa SIM User's Manual\]](#)

[Disabling Timing Checks \[Questa SIM User's Manual\]](#)

tcheck_status

Works in tandem with tcheck_set to report on and enable/disable individual timing checks.

Syntax

```
tcheck_status [-lines] [{-m | -n | -p}] [-r] <instance> [<tcheck>]
```

You must specify the arguments in the order shown.

Description

Prints to 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 disable the reporting of a timing check to prevent the 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 the 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

 Using a tcheck_set or a tcheck_status command on instances compiled with -nodebug produces a warning message.

The tcheck_status command supports using asterisks (*) and ellipses (...) as wildcards for defining hierarchy. An asterisk matches any instance in the current scope. An ellipsis matches any instance in the hierarchy below the current scope. When you use wildcards, all timing checks, up to a maximum of six, must be consecutive, and not intermingled with other arguments. If you are providing more than one timing check, you must place all timing checks in parentheses and double quotes, for example; "(SETUP)". Refer to the Examples section of the [tcheck_set](#) command for descriptions of wildcard usage.

Arguments

- <instance>
(required) Specifies the instance for which to report timing check status.
- -m | -n | -p
(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.

-p — Indicates that the instance is a hierarchical pathname to a port, and that tcheck_set will operate only on the instance that contains the specified port.

- -lines

(optional) Specifies to display the HDL source file and line numbers of the check(s). 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 can 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 3-6 contains a short description of each field in the output.

Table 3-6. 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 not assume it will stay the same between library compiles.
<SDF_Description>	An SDF specification of the timing check including enclosing parentheses ().

Table 3-6. Output Fields for tcheck_status Command (cont.)

Field	Description
<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: <ul style="list-style-type: none"> • MsgON/MsgOFF — violation reporting is enabled/disabled and unchangeable. • MsgOn/MsgOff — violation reporting is enabled/disabled and modifiable.
<XStat>	Violation X generation status indicator: <ul style="list-style-type: none"> • 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 \[Questa SIM User's Manual\]](#)

Time

Used as the suffix for a collection of related commands that allow you to perform comparisons between, operations on, and conversions of, time values for simulation.

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.

Useful when an integer calculation results in a 64-bit value that you need to convert 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) indicating whether 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.

Description

When you do not specify [unit], each of these commands assumes the current simulation time unit, as specified by the Resolution variable in the *modelsim.ini* file, or by the **vsim** -t command. (Refer to the User's Manual for more information on the **Resolution** variable.) For most commands, you can specify units of time (such as ns, us, ps) independently for each <time[1 | 2]>. See the description of each command and examples for more information.

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. 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> — The real number to use as scaling factor. Common values can include:

0.25, 0.5, 1.5, 2, 10, 100

- {+ | -} commas — Controls whether commas display 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

- Entering the command:

```
>ltTime 100ns 1ms
```

Returns:

```
# 1
```

- Entering the command:

```
>addTime {1545 ns} {455 ns}
```

Returns:

```
# 2 us
```

- Entering the command:

```
>gteTime "1000 ns" "1 us"
```

Returns:

```
# 1
```

- Entering the command:

```
>divTime 1us 10ns
```

Returns:

```
# 100
```

- Entering the command:

```
>formatTime +bestunit
```

Returns:

```
# -commas -nodefunit +bestunits
```

- Entering the command:

>scaleTime 3ms 1000

Returns:

3 sec

- Entering the command:

>RealToTime 1.345e04

Returns:

13450 ns

toggle add

Enables collection of toggle statistics for the specified nodes.

Syntax

```
toggle add [-exclude {<list>}] [-countlimit <int>] [-extendedtogglemode 1|2|3] [-full] [-in]
            [-inout] [-internal] [-out] [-ports] [-r] [-unique] [-widthlimit] <node_name>
```

Description

Arguments specified with the toggle add command override those set by the +cover=t(x) arguments set at compile time (vlog/vcom). For example, specifying toggle add -full converts signals previously added as standard (two transition) toggles to extended toggle coverage mode (all six transitions). Similarly, specifying the toggle add command on extended toggle coverage mode toggles (six transitions), converts them to standard coverage toggles (two transitions).

The allowed nodes are:

- Verilog nets and registers
- VHDL signals of type bit, bit_vector, std_ulogic, std_logic, and std_logic_vector
- VHDL Boolean and Integer types (including subranges) and other user-defined Enum types
- SystemVerilog real types

All other types are silently ignored.

You can 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. Refer to [Using the Toggle Add Command](#) in the User's Manual.

You can also collect and view toggle statistics in the Questa SIM GUI. Refer to “[Coverage](#)” in the User's Manual for details.

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. A second toggle add -exclude command issued on the same signal overrides 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. Refer to [ToggleCountLimit](#) in the User's Manual.
- **-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 from 1 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.

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 to enable toggle statistics collection 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 is not 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. (Refer to [ToggleCountLimit](#) in the User's Manual.)
- **<node_name>**
(required) Enables toggle statistics collection for the named node(s). Accepts multiple names and wildcards.

Return Values

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 \[Questa SIM User's Manual\]](#)

[Toggle Exclusion Management" \[Questa SIM User's Manual\]](#)

[toggle disable](#)

[toggle enable](#)

[toggle report](#)

[toggle reset](#)

toggle disable

Disables toggle coverage statistics collection on the specified nodes.

Syntax

```
toggle disable [-all] | [-in] [-out] [-inout] [-internal] [-ports] [-r] <node_name>
```

Description

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, refer to [Exclude Nodes from Toggle Coverage](#) in the User’s Manual.

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 to disable toggle statistics collection 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). Accepts multiple names and wildcards.

Related Topics

[Toggle Coverage \[Questa SIM User's Manual\]](#)

[toggle add](#)

toggle disable

[toggle enable](#)

[toggle report](#)

[toggle reset](#)

toggle enable

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

[Toggle Coverage \[Questa SIM User's Manual\]](#)

[Toggle Exclusion Management \[Questa SIM User's Manual\]](#)

[toggle add](#)

[toggle disable](#)

[toggle report](#)

[toggle reset](#)

toggle report

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.

Note

 To ensure that you are reporting all signals in the design, use `toggle report -duplicates`.

Syntax

```
toggle report [-all] [-duplicates] [-file <filename>]  
    [-select {inputs | outputs | inout | ports | internals}]  
    [-instance <path> [-recursive]] [-onexit] [<signal>...] [-showambiguity] [-summary] [-top]  
    [-verbose]
```

Description

The command also displays a summary of the number of nodes checked, the number that toggled, the number that did not toggle, and a percentage that toggled.

You can also collect and view toggle statistics in the Questa SIM GUI. Refer to “[Coverage](#)” in the User’s Manual for details.

To get the toggle report, you must:

- Enable statistics collection with the [toggle add](#) command.
- Run the simulation with the [run](#) command.
- Produce the report with the `toggle report` command.

Ordering of toggle nodes

The ordering of nodes in the report depends on how you specify the signal list. If you use a wildcard in the signal argument (for example, `toggle report -all -r /*`), the nodes are listed in the order signals are found when searching down the context tree. Multiple elements of the same net are listed multiple times. If you do not use a wildcard, the nodes are listed in the order they were originally added to toggle coverage, and elements are not duplicated.

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 displays 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 to report toggle statistics recursively into subregions. Omitting -recursive limits toggle statistic reporting 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 for which to display toggle statistics. You can specify multiple signal names, separated by spaces. Allows wildcards.
- **-select {inputs | outputs | inout | ports | internals}**
(optional) Reports on input, output, inout, all ports, or internal signals.
- **-showambiguity**
(optional) Specifies to report 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. The default is to use the name of the wildcard-matching segment.
- **-unique**
This option is obsolete with version 6.3. By default, toggles are always unique.
- **-verbose**
(optional) Specifies to include all values taken on by integer variables.

Related Topics

[Toggle Coverage \[Questa SIM User's Manual\]](#)

[toggle add](#)

[toggle disable](#)

[toggle enable](#)

[toggle reset](#)

toggle reset

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). Accepts multiple names and wildcards.

Related Topics

[Toggle Coverage \[Questa SIM User's Manual\]](#)

[toggle add](#)

[toggle disable](#)

[toggle enable](#)

[toggle report](#)

tr color

Changes the color scheme of individual transactions and entire streams, either in a specific wave window or for all wave windows.

Note

 This is the command equivalent of the Colors tab in the Transaction-Stream Properties dialog.

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>]
```

Description

The option <color> in the arguments specifies the color to use. You can use either a standard X Window color name or an RGB value (for example, #357f77); You must enclose multi-word names in quotes; for example, (“light blue”).

If you do not specify any arguments, this command returns the value of each configuration item in a Tcl list.

Unique abbreviations are accepted for all arguments.

Arguments

- **-stream <stream> [<stream> ...]**
(required unless specifying -transaction.) Specifies that all objects in the tr color command are transaction streams. <stream> is the path to a transaction stream. Allows multiple streams, 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 with the “[tr uid](#)” command.
You can specify <uid> either with the 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 you use the full UID, you must surround it with curly braces ({}).
You can use multiple ID specifications, 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. Accepts either a standard X Window color name or an RGB value (for example, #357f77); multi-word names must be in quotes; for example, (“light blue”).
- **-default**
(optional) Removes any color overrides on the specified streams or transactions. This option takes precedence over any other option that sets color.
- **-get**
(optional) Specifies to return a list of the color schemes for each transaction or stream. If the command changes any colors, this argument returns the resulting color scheme. Each scheme is 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 name.
- **-nametext <color>**
(optional) Select the color for the transaction name.
- **-win <wave>**
(optional) Specifies the wave window to which the changes should apply. <wave> is the Tk name (not the title) of 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 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 \[Questa SIM User's Manual\]](#)

[tr uid](#)

[tr order](#)

tr order

Controls which attributes are visible and the order in which they appear.

Syntax

```
tr order [[-attributes <attrs>] [-default] [-win <wave>] <stream> ...] |  
[-hidden] [-visible] [-win <wave>] {<stream> ...}]
```

Description

The tr order command 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).

When two streams have different attributes, or the same attributes in a different order, this command resolves the differences while 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 that do not match 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 differ from that of another stream in the command line.

All arguments accept unique abbreviations.

Arguments

- **-attributes <attrs>**
(optional) Specifies that the attributes for the specified stream(s) are made 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 you list more than one attribute, you must enclose the list in curly braces “{ }”. Use an empty list of attributes (“{ }”) to specify that there be no visible attributes.
- **-default**
(optional) Removes any visibility and order overrides on the specified streams or transactions. This option takes precedence over –attributes.
- **-hidden**
(optional) Returns a list of the hidden attributes for each specified stream or transaction. If -visible is set, the hidden attributes are in a list following the visible attributes.

- **-visible**
(optional) Returns 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) 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) Specifies the path to a transaction stream. You can enter multiple streams as a space-separated list. You cannot use wildcards. The **tr order** command requires either a stream or a 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 \[Questa SIM User's Manual\]](#)

[tr color](#)

[tr uid](#)

tr uid

Returns a list of unique transaction IDs for a specified time span on specified streams, or for all times on specified streams.

Note

 A transaction UID is the logical name of its dataset; a 64-bit serial number created during simulation.

Syntax

```
tr uid [-time <time> <stream> ...] | [-start <time> -end <time> <stream> ... ] | [<stream> ...]
```

Description

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, from the start of the span to 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)
- a listing of transactions that are active at one specific time (using -time)

All arguments accept unique abbreviations.

Arguments

- -end
(required in conjunction with the -start option) Specifies the start of the span of time during which to obtain UIDs.
- <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 during which to obtain UIDs.
- -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. You can specify more than one stream. You cannot use wildcards.

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 \[Questa SIM User's Manual\]](#)

[tr color](#)

[tr order](#)

transcribe

Displays a command in the Transcript window, and then executes the command.

Syntax

`transcribe <command>`

Description

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. You can also use the transcribe command to force a command to be echoed.

Returns nothing.

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
```

transcript

Controls echoing of commands executed in a macro file. Reports the current setting when entered with no options.

Syntax

transcript [on | off | -q | quietly]

Arguments

- **on**
(optional) Specifies that commands in a macro file are echoed to the Transcript window as they are executed.
- **off**
(optional) Specifies that commands in a macro file are not 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. For information about turning off echoing for individual commands, see the [quietly](#) command.

Examples

- Echo commands within a macro file to the Transcript window as they are executed:

transcript on

- If issued immediately after the previous example, the command:

transcript

returns

Macro transcripting is turned ON.

Related Topics

[Transcript Window \[Questa SIM GUI Reference Manual\]](#)

[transcribe](#)

transcript file

Sets or queries the current PrefMain(file) Tcl preference variable.

Syntax

transcript file [<filename>]

Arguments

- <filename>

(optional) Specifies a name for the transcript file. Allows wildcard characters, and “stdout” or “stderr” are valid file names. Specifying a new file closes the existing transcript file, and opens a new transcript file. Specifying an empty string (“”) closes the existing file, and no new file is opened. If you do not specify this argument, the current filename is returned.

Note

 To prevent overwriting older transcript files, include 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.

Description

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.

Examples

- Close the current transcript file and stop 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*.

- Create a transcript containing only data from the second millisecond of the simulation.

```
transcript file ""
run 1 ms
transcript file transcript
run 1 ms
```

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.

Related Topics

- ["Creating a Transcript File" \[Questa SIM User's Manual\]](#)
- [“Setting GUI Preferences” \[Questa SIM GUI Reference Manual\]](#)
- [Transcript Window \[Questa SIM GUI Reference Manual\]](#)
- [transcript path](#)
- [transcript sizelimit](#)

transcript path

Returns the full pathname to the current transcript file.

Syntax

`transcript path`

Arguments

None

Related Topics

["Creating a Transcript File" \[Questa SIM User's Manual\]](#)

[“Setting GUI Preferences” \[Questa SIM GUI Reference Manual\]](#)

[“Transcript Window” \[Questa SIM GUI Reference Manual\]](#)

[transcript file](#)

transcript sizelimit

Sets or queries the current preference value for the transcript fileSizeLimit value.

Syntax

`transcript sizelimit [<size>]`

Arguments

- <size>

(optional) Specifies the size, in KB, of the transcript file. The default is 0 or unlimited. The actual file size can 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

 If the size limit is reached, the transcript file is saved and simulation continues. You can set the size of the transcript file with the \$PrefMain (fileSizeLimit) Tcl variable in the Preferences dialog. Refer to “[Setting GUI Preferences](#)” in the GUI Reference Manual for more information.

Related Topics

[“Creating a Transcript File” \[Questa SIM User's Manual\]](#)

[“Setting GUI Preferences” \[Questa SIM GUI Reference Manual\]](#)

[“Transcript Window” \[Questa SIM GUI Reference Manual\]](#)

[transcript file](#)

transcript wrapcolumn

Defines the column width when wrapping output lines in the transcript file.

Syntax

```
transcript wrapcolumn <integer>
```

Arguments

- <integer>

An integer that defines the width, in characters, before forcing a line break. The default value is 30000.

Description

The wrap occurs at the first white-space character after reaching the transcript wrapwscolumn value, or at exactly the column width if no white-space is found.

transcript wrapmode

Controls wrapping of output lines in the transcript file.

Syntax

`transcript wrapmode [0 | 1 | 2]`

Arguments

- 0
(default) Disables wrapping.
- 1
Enables wrapping, based on the value of the transcript wrapcolumn command, which defaults to 30,000 characters.
- 2
Enables wrapping and adds a continuation character (\) at the end of every wrapped line, except for the last.

transcript wrapwscolumn

Defines the column width when wrapping output lines in the transcript file.

Usage

```
transcript wrapwscolumn <integer>
```

Arguments

- <integer>

An integer specifying that the wrap will occur at the first white-space character after reaching the specified number of characters. If there is no white-space, the wrap occurs at the transcript wrapcolumn value. The default value is 27000.

triage dbfile

Creates a database (*questasim.tdb*) of test data and/or simulation messages extracted from a UCDB, a WLF file, or a LOG file.

Note

 You can run this command from either the shell command line or the vsim command prompt.

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>]
    [-msgsfile <msgs-filename> [...] [-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]
    [[-wlffattr <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]
```

<txoptions>

Arguments for importing transactions from a WLF file.

```
--tx [-testname <testname>] <wlf_filename>
```

Description

By default, the triage database file (TDB) is cumulative: it preserves tests and/or messages added by previous commands. You can use the -clear switch to remove existing data before adding the extracted data into the TDB.

The command does not insert messages into the database that are associated with tests already represented in the database, unless you specify -append or -force.

The triage dbfile command imports user-defined attributes from the test data record(s) in a UCDB file into the TDB file. You can display these attributes as columns in the Results Analysis window and use them in hierarchy configurations and filter expressions. The text report (the [triage report](#) command) does not support these attributes. To support the user-defined attribute table, the [triage query](#) command "-attributes" option allows direct SQL queries to use the "(%ATTRIBUTES%)" token to refer to the attributes table. Also, when the triage dbfile command imports a UCDB, the status message reports the number of user-defined attributes added to the TDB file.

The triage dbfile command attempts to find the WLF (or plain-text LOG) file associated with each test it extracts from a UCDB file. The command uses the WLFNAME test data record attribute (or the attribute specified by -wlffattr) to locate the relevant WLF file. If no WLF file is specified (or if the WLF file cannot be found), the command uses the LOGNAME test data record attribute (or the attribute specified by -logattr) to locate a plain-text LOG file containing messages from the simulation. If the command finds a WLF file, it then disregards any specified LOG file as the source for import.

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.

You can associate each imported message with a "testname". If you are not using a UCDB file, you must use the -testname argument to associate a testname with the data. With -testname, you can specify either a WLF file or a plain-text LOG file for import (not both). If you do not use -testname, any number of WLF and/or LOG files may be imported, in which case the basename of each file (that is, 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 are automatically 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. Refer to [“Message Transformation”](#) in the Verification Management User’s Manual for further information.

Arguments

- `-append`
(optional) Appends new data to any existing data associated with a specified testname.

- **-buflines <int>**
(optional) Determines the number of input LOG (.log) file lines in the buffer at any given time; relevant when using a transformation rulesfile. Controls the maximum number of lines a single message can occupy and still be detected. Default is 10.
- **-clear**
(optional) Clears data from an existing database. If you do not invoke -clear, the default is to 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 must contain only 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.
- **<logfile> [<logfile>]...**
(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 you run the command from the vsim prompt, the tdb_filename you specify with -name becomes "sticky" in that future commands you apply will use that filename, until you explicitly set it to another name. For commands issued at the command shell level, previous -name settings are not preserved.

- **-nowrite**
(optional) Suppresses the insertion of the tests and/or messages into the database. Transformation processing still occurs. You can use -nowrite 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.
- **-msgsfile <msgs-filename> [. . .]**
Inserts already transformed messages into the triage database. Stores each message in the file as a key-value list. (The key corresponds to the fieldname – which can be a predefined field or a user-defined field – while the value is the content inserted into the database.) A key msg is necessary for a successful import of the message into the triage database.
- **-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 you include the -rulesfile argument 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.
You can have multiple rules files: 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 that 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 also dropped.
- **-severity {I|F|E|W|N}**
(optional) The default is “-severity IFE”, which specifies to save only internal error, fatal and error messages. Only WLF messages that match the specified severity are inserted into the triage database.
 - I — internal error
 - F — fatal, failure
 - E — error
 - W — warning
 - N — note
- **-severityAll**
(optional) The same as -severity IFEWN.

- **-tx [-testname <testname>] <wlffile>**
(optional) Specifies to import the transactions from the specified WLF file. If you do not specify the -testname argument, it is determined from the UCDB, or extracted from the input filename.
- **-testname <testname> {<wlffilename> | <logfilename>}**
(optional) Used to override other methods of determining the testname for message(s). Specifying -testname allows only one WLF and/or plain-text LOG input files. If you do not provide the -testname argument, 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 the testname.
- **-teststatus {F|E|W|P|M|S}**
(optional) Specifies one or more test status codes to control the automatic import of a WLF or LOG file associated with each test. If WLF files are read using "wlffattr", such as test attribute WLFNAME, then the test status is checked first before proceeding. With the default setting of -teststatus FE, WLF messages load into the database only for tests with a fatal or error status.
 - F — fatal
 - E — error
 - W — warning
 - P — pass or OK
 - M — merge error
 - S — missing
- **-teststatusAll**
(optional) 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).
- **-wlffattr <str>**
(optional; applies only to UDCB files) Specifies 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 "-wlffattr" 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

- You store 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 -wlffattr 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 \[Questa Verification Management User's Manual\]](#)

[triage dbinsert](#)

[triage report](#)

[triage query](#)

[triage view](#)

triage dbinsert

Places one or more messages directly into a triage database (TDB) file.

Syntax

```
triage dbinsert [-verbose] [-debug] [-name <tdb_filename>] [-nowrite]
[-rulesfile <rulesfilename> [-rulesfile <rulesfilename2>]]
[-setseverity <string>] [-setttime <int_time>] [-severity {I|F|E|W|N}] [-severityAll]
[-testname <testname>] <message> [<message>]...
```

Description

When inserted, the messages are handled in the same way as a single message from a WLF or plain-text LOG file:

- Each message is transformed if a transform rules file is specified with the -rulesfile option.
- The severity is checked against the specified severity filter (-severity).
- 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.

You can run this command from either the shell command line or from the vsim prompt.

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 insert. 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. Use this switch 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. Allows the use of multiple rules files: 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. Specify the time as an integer followed immediately by one of the recognized time unit identifiers. You can use a single optional underscore 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 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 is not imported into the database. Default behavior is -severity IFEWN (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 \[Questa Verification Management User's Manual\]](#)

[triage dbfile](#)

[triage report](#)

[triage query](#)

[triage view](#)

triage dbrefresh

Clears the old triage database, and re-creates the database using a saved history of previous "triage dbfile" and "triage dbinsert" commands.

Syntax

triage dbrefresh [[-name <tdb_filename>](#)] [[-verbose](#)]

Arguments

- [-name <tdb_filename>](#)

Optional. Specifies a path and file name for the *.tdb* file to refresh. Default filename is *questasim.tdb* located in the current directory.

- [-verbose](#)

Optional. Prints additional information to the screen.

Description

You can run this command from either the command shell or the vsim command prompt.

Related Topics

[Analyzing Verification Results \[Questa Verification Management User's Manual\]](#)

[triage dbfile](#)

[triage dbinsert](#)

[triage query](#)

[triage view](#)

[triage report](#)

triage passfail

Extracts messages from logfiles and the wlffile with the aid of the transform rules file.

Syntax

```
triage passfail [-debug] [-file <file>] [-verbose] [<inputoptions>] [<testoptions>]  
[<msgoptions>] [-help]
```

where:

<inputoptions>

```
[-continue] [-inputsfile <file>] [-rulesfile <rulesfilename>] [-rulesfile <rulesfilename2>]...]  
[-buflines]
```

<testoptions>

```
[-teststatus {F|E|W|P|M|S}] [-teststatusAll] [-wlffattr <str>] [-logattr <str>] [-ignorewlf]  
[<ucdbfilename> [<ucdbfilename>]]
```

<msgoptions>

```
[<logfilename> [<logfilename>]...] [-severity {I|F|E|W|N} | -severityAll] -testname <testname>  
[<wlffilename>]
```

Arguments

- **-buflines**
(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 can occupy and still be detected.
- **-continue**
(optional) Continues transformation until the end of the input files. Useful for debugging.
- **-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.
- **-file <file>**
(optional) Stores extracted messages in the designated <file>, with corresponding fields.
- **-ignorewlf**
(optional) Forces the triage passfail command to import simulation messages from the LOG file instead of from the WLF file. The default behavior is for triage passfail to give preference to the WLF file if valid message data is found, and use the LOG file only if valid message data is missing from the WLF file. Effective only when importing a UCDB file.

- **-inputsfile <file>**
(optional) Specifies the path to a file containing a list of .wlf, and/or plain text LOG files. The specified file must contain one input file per line, each of which can include path variables, such as: \$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, the value applies to every UCDB on the same command line.
- **<filename> [<filename>]...**
(optional) Specifies the name and/or path to a plain-text LOG file. You can use multiple logfiles for the pass/fail determination of a test.
- **-rulesfile <rulesfilename> [-rulesfile <rulesfilename2>]...**
(required only for transforming plain-text LOG files; optional otherwise) Specifies a transformation rules file (applies only to .wlf or .log files). If the command line includes the -rulesfile argument, 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. You can include TCL commands and variables in rules files. You can include multiple rules files. 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 processed 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 also dropped.
- **-severity {I|F|E|W|N}**
(optional) Default behavior is "-severity IFE" whereby only internal error, fatal and error messages produces a failure.
 - I — internal error
 - F — fatal, failure
 - E — error
 - W — warning
 - N — note
- **-severityAll**
(optional) The same as -severity IFEWN.
- **-testname <testname>**
(optional) Used to override other methods of determining the testname for message(s). If the -testname argument is not provided, the testname is determined from the UCDB.

- **-teststatus {F|E|W|P|M|S}**
(optional) Specifies one or more test status codes to use 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 "wlffattr" – such as test attribute WLFNAME – then the test status is checked first before proceeding. Using the default settings of -teststatus FE, only tests that had a fatal or error status would have corresponding WLF messages loaded into the database.
- **-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).
- **-verbose**
(optional) Prints additional information to the log output.
- **-wlffattr <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.
- **<wlffilename>**
(required if neither UCDB nor LOG file is specified; optional otherwise) Specifies the name and/or path of the WLF file name.

Description

This command exits with an exit code of 95 if any of the extracted messages matches the severity filter. The input files are assumed to be related to a single test. You can use the -file filename option to optionally store the extracted messages in a text file.

triage query

Provides programmatic access to the contents of the triage database (TDB) for debug and scripting purposes.

Syntax

```
triage query [-help] [-name <tdb_filename>] <options>
```

<options>

```
<options> = -analyze | <sql_options> | <query_options>
```

<sql_options>

```
<sql_options> = -statement <query_string> [-elapsed]
```

<query_options>

```
<query_options> = [-files] [[-unique | -verbose] <table_options>]
```

<table_options>

```
<table_options> = [-attributes] [-fields] [-history] [-messages] [-tests] [-properties]
```

Description

The command has two forms:

- You can use -statement to specify one or more SQL query statements 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 use aliases to specify certain tables within the TCL code. See [Table 3-7](#) for a list of alias names.
- If you do not specify a query with -statement, the triage query command returns the contents of any specified tables of data from the TDB. If you do not specify any tables, the content of all tables is returned. Each table's contents are in the "list-of-lists" format, and the returned result includes all fields in each table. When data is returned from multiple tables, it is shown in the order listed in [Table 3-7](#).

Table 3-7. 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

You can run this command from either the shell command line or the vsim command prompt.

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 (that is, 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 (that is, the contents of the "properties" table).
- **-statement <query_string>**
Optional. Specifies an SQL query string to be applied to the database. This argument can appear multiple times in a single command. If you specify multiple -statement arguments, the triage query command returns a list of result tables in the order in which the query statements appear 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 (that is, a list of imported UCDB test data records).

- **-unique**

Optional. Returns a list of unique values for the main field in that table (for example, 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 \[Questa Verification Management User's Manual\]](#)

[triage dbfile](#)

[triage dbinsert](#)

[triage dbrefresh](#)

[triage report](#)

[triage view](#)

triage report

Reads in a triage database (.tdb) and prints out a text report, after applying specified filters.

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. If you do not specify -append, 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 you use the -details argument.

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 the output filename. If you do not specify -file, 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 you use the -details argument.
- **-nummsg <int>**
(optional) Limit the maximum number of the same message to the specified number. The default behavior of “1” compresses all identical messages into one message and prints the earliest time. The value of <int> must be a positive integer (it ignores “0” and generates a warning). When <int> is an integer greater than 1, the same message prints out the specified number of times, and 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

Test record information is printed for matches to the filtering. Test record information is displayed only when you also specify -details T.

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.

Description

You can run this command from either the shell command line or the vsim command prompt. The triage report lists messages from highest level of severity to lowest: messages with fatal or error severity are shown first, so that you can more easily focus on highest priority failures.

Related Topics

[Analyzing Verification Results \[Questa Verification Management User's Manual\]](#)

[triage dbfile](#)

[triage dbinsert](#)

[triage dbrefresh](#)

[triage query](#)

[triage view](#)

triage view

Loads a triage database (*.tdb*) into the Verification Results Analysis window.

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 use in the new window.
- **-filterexp <name>**
(optional) Specifies the previously named and saved filtered expression to use in the new window.
- **-hierconfig <name>**
(optional) Specifies the previously named and saved hierarchy configuration to use in the newly opened window.
- **-name <tdb_filename>**
(optional) Specifies the name of the triage database to open with the command. If you do not specify -name, 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 Results Analysis window's Sort Configuration dialog box or pulldown list.

Description

You can invoke this command only 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. You can have multiple Results analysis windows open at the same time.

Related Topics

[Analyzing Verification Results \[Questa Verification Management User's Manual\]](#)

[triage dbfile](#)

[triage dbinsert](#)

[triage dbrefresh](#)

[triage query](#)

tssi2mti

Converts a vector file in TSSI Format into a sequence of force and run commands.

Syntax

```
tssi2mti <signal_definition_file> [<sef_vector_file>]
```

Description

This command writes the stimulus to the standard output.

The source code for tssi2mti is provided in the *examples* directory as:

```
<install_dir>/examples/tssi2mti/tssi2mti.c
```

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 convert. If no file is specified, standard input is used.

Examples

- The following command produces 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

[run](#)

typespec

Queries class names and class relationships of SystemVerilog classes.

Syntax

```
typespec <pattern> [-dataset <value>] [-isa <value>] [-class <value> | -ancestry <value>]  
[-exact] [-regexp] [-tcl] [-indent <value>]]
```

Arguments

- <pattern>
(optional) Specifies the pattern to query for.
- -dataset <value>
(optional) Identifies a specific dataset.
- -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 to treat all <pattern> arguments as regular expressions.
- -tcl
(optiona) Returns the results in as a Tcl list.
- -indent <value>
(optional) Prefixes each level of the class inheritance hierarchy with <value>.

Description

You can use this command and the find insource command for debugging when you encounter design elaboration errors.

Related Topics

[DISABLE_ELAB_DEBUG \[Questa SIM User's Manual\]](#)

ui_VVMode

Specifies the actions to take when encountering UI registration calls used by verification packages. Returns the current setting when specified without an argument.

Syntax

`ui_VVMode [full | logclass | logobj | nolog | off]`

Description

UI registration calls, which are Verilog system tasks specific to Questa SIM, are typically included in verification packages to make key information about the packages 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`.

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.

unsetenv

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 to delete.

Related Topics

[setenv](#)

[printenv](#)

up

Searches for object transitions or values in the specified List window.

Syntax

```
up [-expr {<expression>}] [-falling] [-noglitch] [-rising] [-value <sig_value>]  
[-window <wname>] [<n>]
```

Description

The up command executes the search on the objects that are currently selected in the window, starting at the time of the active cursor. The active cursor moves to the found location.

You can use this command to:

- Move to consecutive transitions.
- Find the time at which an object takes on a particular value.
- Find the time that an expression of multiple objects evaluates to true.

Refer to the [down](#) command for related functionality.

Usage

To use the up command:

1. Click the desired waveform.
2. Click the desired starting location.
3. Run the up command. (The [seetime](#) command can initially position the cursor from the command line, if desired.)

Returns: <number_found> <new_time> <new_delta>

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 can involve more than one signal, but is limited to signals that have been logged in the referenced Wave window. You can specify a signal either by its full path or by the shortcut label displayed in the Wave window.

{<expression>} — Any valid expression. See “[GUI_expression_format](#)” on page 42 for the format of the expression. You must enclose the expression in 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, this option is ignored.

- **-noglitch**
Specifies to ignore delta-width glitches.
- **-rising**
(optional) Searches for a rising edge on the specified object if that object is a scalar. If it is not a scalar, this option is ignored.
- **-value <sig_value>**
(optional) Specifies a value of the signal to match. You can select only one signal, but that signal can be an array.

`<sig_value>` — Must be specified in the same radix used to display the selected waveform. 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, uses the default Wave window. 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, returns the number found with a warning message, and positions the marker at the last match.

Examples

- Find the last time at which the selected vector transitions to FF23, ignoring glitches.

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](#)” on page 42.

- Search up for an expression that evaluates to a boolean 1 when signal *clk* just changed from low to high, 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* equal 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

[seetime](#)

uvm call

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all -classdebug`.
- Elaborate the UVM testbench by executing the following command:

run 0

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 enter `uvm` on the command line to find additional information about the group of commands. For help with a particular subcommand, enter `uvm help <subcommand>`. Refer to the Related Topics for other subcommands.

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. You can specify the path 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 available only after the design has been elaborated with a `run 0` or after running the design. Refer to [The Class Instance Identifier](#) in the User's Manual 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 specified in the following format: `<string>.<string>.<string>....<function_name>`. For example, `uvm_test_top.middle.bottom0.my_obj`, which 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, 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 looks first into the current context to find the specified function. If the function does not exist in the current context, then the argument string is parsed to determine if it contains one of the hierarchy specifications described above. In all cases the command will check first to see if the path contains a CIID string. If it does not, then by default it will scan the path as a UVM-style path to the function. You can change the default behavior 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) Specify all arguments required by the function in a space-separated list in declaration order. If a function has default arguments, you can omit the arguments 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. You must enclose any string containing spaces or special characters 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 you want a UVM-style pathname, you can use the uvm uvmpath subcommand. If a function has type inout, out, or ref arguments, you must pass in suitable user design objects as arguments. Any passed in argument is first tested to determine if it is an appropriate constant value. If it is not, then the argument is 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 is given precedence. If you want the design object instead, you can supply the full hierarchical path to the object 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

Prerequisites:

Before using this command, do the following:

- Simulate with [vsim -uvmcontrol=all](#)

Enables or disables printing of configuration database reads and writes to the transcript as they occur.

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 the Related Topics for links to the other subcommands.

Syntax

`uvm configtracing {0 | 1}`

Description

UVM configuration database reads and writes typically occur during the build phase of UVM elaboration, so you should turn `uvm configtracing` on or off before UVM elaboration happens, which is before you issue a "run 0". If specified without arguments, `uvm configtracing` returns the current tracing mode.

Arguments

- 0
(default) Disables config database and resource database tracing.
- 1
Enables config database and resource database tracing.

You can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to "[Command Shortcuts](#)" on page 38 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

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all -classdebug`.
- Elaborate the UVM testbench by executing the following command:

run 0

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 the Related Topics for links to the other subcommands.

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. You can specify the path 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 available only after you elaborate the design with a `run 0` or run the design. Refer to [The Class Instance Identifier](#) in the User's Manual for more information.

Note

 A CIID is unique for a given simulation. If you modify the design, or run the same design with different parameters, randomization seeds, or other configurations that change the order of operations, the class instance may change. For example, `@packet@134` in one simulation run may not be the same `@packet@134` in another simulation run where the design has changed.

UVM Hierarchical Path — Specify the UVM Hierarchical Path (also called the UVM path) in the following format: `<string>.<string>.<string>....<class_type>`. An example is `uvm_test_top.middle.bottom0.my_obj.`, the path registered for the object when you allocate the class with the UVM `create()` method. A UVM `get_full_name()` function returns the name of this path.

Simulation Path — Specify the Simulation Path, which is the hierarchical path mapping for the UVM object within the sim:/uvm_root context, 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 match the simulator's path delimiter settings, and component names that do not conform to Verilog naming rules may be changed to use Verilog escaped identifiers.

The command looks first into the current context to find the specified object and, if the object does not exist in the current context, it parses the argument string looking for one of the hierarchy specifications described above. In all cases, the command checks first to see if the path is a CIID string, and if it is not, then by default the command scans the path as a UVM-style path name. You can use the [uvm mapmode](#) command to change the default behavior to be simulation-style path names. Finally, even when the path name mapping mode is in the default UVM-style mode, the command can still accept simulation-style paths if they start with a *sim:/uvm_root* prefix.

In addition to the full command syntax, you can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to “The Class Instance Identifier” 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

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all -classdebug`.
- Elaborate the UVM testbench by executing the following command:

run 0

Lists the set of HDL registers attached to the UVM register models in a design.

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 the Related Topics below for links to the other subcommands.

Syntax

`uvm findregisters [-acc] <uvm_reg_block_path>`

Description

You can use the list of HDL registers to create a file that allows visibility into the register blocks after optimization.

Arguments

- `<uvm_reg_block_path>`
(required) Specifies the path to a UVM register block. You can specify the path in the following ways:
 - UVM Path — (default) Specify the UVM Path 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) Specify the Simulation Path, which is the hierarchical path mapping for the UVM object within the `sim:/uvm_root` context, in the following format: `sim:/uvm_root/<string>/.../ <uvm_component>`. For example, `sim:/uvm_root/uvm_test_top/middle/bottom0/my_obj`. The hierarchical simulation path is similar to a UVM Hierarchical Path, except that path delimiters match the simulator's path delimiter settings, and component names that do not conform to Verilog naming rules may be changed to use Verilog escaped identifiers
- `-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 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

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all -classdebug`.
- Elaborate the UVM testbench by executing the following command:

run 0

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 the Related Topics for links to the other subcommands.

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) Specify the Simulation Path, which is the hierarchical path mapping for the UVM object within the `sim:/uvm_root` context, 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 match the simulator's path delimiter settings, and component names that do not conform to Verilog naming rules may be changed to use Verilog escaped identifiers.

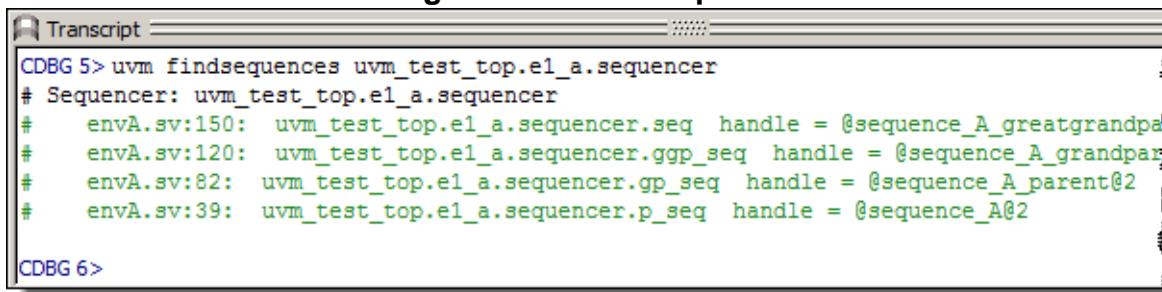
Examples

- List all active sequences for the sequencer `uvm_test_top.e1_a.sequencer`.

uvm findsequence uvm_test_top.e1_a.sequencer

Returns a list of hypertext links for the active sequences under the specified sequencer (Figure 3-2).

Figure 3-2. UVM Sequences



The screenshot shows a transcript window with the following text:

```
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

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all -classdebug`.
- Elaborate the UVM testbench by executing the following command:

run 0

Returns the simulation handle (class instance identifier or CIID) of the specified UVM object 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 the Related Topics for links to the other subcommands.

Syntax

`uvm handle <uvm_object_path>`

Description

This command returns 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.

Arguments

- `<uvm_object_path>`
(required) A UVM object name, including a path if the object is not in the current context. You can specify the path in the following ways:

UVM Hierarchical Path — Specify the UVM Hierarchical Path (the UVM path) in the following format: `<string>.<string>.<string>....<class_type>`. For example, `uvm_test_top.middle.bottom0.my_obj` is the path registered for the object when you use the UVM `create()` method to allocate the class. A UVM `get_full_name()` function call returns the name of this path.

Simulation Path — Specify the Simulation Path, the hierarchical path mapping for the UVM object within the `sim:/uvm_root` context, 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 match the simulator's path delimiter settings, and component names that do not conform to Verilog naming rules may be changed to use Verilog escaped identifiers.

The command first looks for the specified object in the current context. If the object does not exist in the current context, the command parses the argument string for one of the

hierarchy specifications described above. By default, the command scans the path as a UVM-style path name. You can use the [uvm mapmode](#) command to change the default behavior to be simulation-style path names. Even when path name mapping is in the default UVM-style mode, it 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

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all -classdebug`.
- Elaborate the UVM testbench by executing the following command:

run 0

Sets the UVM path mapping mode for UVM commands.

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 the Related Topics or links to the other subcommands.

Syntax

`uvm mapmode [0 | 1]`

Description

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, the command displays the current mapping setting.

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`.
You can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to “[Command Shortcuts](#)” on page 38 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

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all -classdebug`.
- Elaborate the UVM testbench by executing the following command:

run 0

Returns configuration information matching a specified component and/or access information for a specified configuration object variable 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 the Related Topics for links to the other subcommands.

Syntax

`uvm printconfig [<uvm_component_path> [<config_obj_name>]]`

Description

If you do not specify an argument, this command returns the global configuration table to the transcript.

Arguments

- `<uvm_component_path>`
(optional) Specifies the path to a UVM object. You can use the asterisk wildcard (*) to match all components. You can specify the path in the following ways:
 - UVM Path — (default) Specify the UVM Path 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) Specify the Simulation Path, which is the hierarchical path mapping for the UVM object within the `sim:/uvm_root` context, 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 match the simulator's path delimiter settings, and component names that do not conform to Verilog naming rules may be changed to use Verilog escaped identifiers.
- `<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.

You can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to “[Command Shortcuts](#)” on page 38 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

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all -classdebug`.
- Elaborate the UVM testbench by executing the following command:

run 0

Returns UVM global factory information to the transcript. 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 the Related Topics for links to the other subcommands.

Syntax

`uvm printfactory`

You can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to “[Command Shortcuts](#)” on page 38 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

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all -classdebug`.
- Elaborate the UVM testbench by executing the following command:

run 0

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 the Related Topics for links to the other subcommands.

Syntax

`uvm printstreams [<uvm_component_path>]...`

Arguments

- `<uvm_component_path>`

(optional) Specifies a UVM object. You can use the asterisk wildcard (*) to match all components. Specify the path in the following ways:

UVM Path — (default) Specify the UVM Path 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) Specify the Simulation Path, which is the hierarchical path mapping for the UVM object within the `sim:/uvm_root` context, 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 match the simulator's path delimiter settings, and component names that do not conform to Verilog naming rules may be changed to use Verilog escaped identifiers.

You can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to “[Command Shortcuts](#)” on page 38 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

Prerequisites:

Before using this command, do the following:

- Simulate with **vsim** -uvmcontrol=all -classdebug.
- Elaborate the UVM testbench by executing the following command:

run 0

Returns the UVM global testbench topology to the transcript. 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 the Related Topics for links to the other subcommands.

Syntax

uvm printtopology

Note

You can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to “[Command Shortcuts](#)” on page 38 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

Prerequisites:

Before using this command, do the following:

- Simulate with [vsim -uvmcontrol=all](#).

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 the Related Topics for links to the other subcommands.

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. You can use the asterisk wildcard (*) to match all components. You can specify the path in the following ways:

UVM Path — (default) Specify the UVM Path 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) Specify the Simulation Path, which is the hierarchical path mapping for the UVM object within the *sim:/uvm_root* context, 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 match the simulator's path delimiter settings, and component names that do not conform to Verilog naming rules may be slightly changed to use Verilog escaped identifiers.

You can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to “[Command Shortcuts](#)” on page 38 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 sympath

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all`.
- Elaborate the UVM testbench by executing the following command:

run 0

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 the Related Topics for links to the other subcommands.

Syntax

`uvm sympath <uvm_component_path>`

Arguments

- `<uvm_component_path>`
(required) Specifies a UVM object path name.

UVM Path — Specify the UVM Path in the following format:
`<string>.<string>.<string>....<uvm_component>`. For example,
`uvm_test_top.middle.bottom0.my_obj`.

You can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to “[Command Shortcuts](#)” on page 38 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 sympath 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

Prerequisites:

Before using this command, do the following:

- Simulate with `vsim -uvmcontrol=all`.
- Elaborate the UVM testbench by executing the following command:

run 0

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 the Related Topics for links to the other subcommands.

Syntax

`uvm traceobjections <uvm_component_path> [0 | 1]`

Arguments

- `<uvm_component_path>`
(required) Specifies a UVM object. You can use the asterisk wildcard (*) to match all components. Specified without the second argument, returns the current trace objection setting for the specified component to the transcript.

You can specify the path in the following ways:

UVM Path — (default) Specify the UVM Path 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) Specify the Simulation Path, which is the hierarchical path mapping for the UVM object within the `sim:/uvm_root` context, 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 match the simulator's path delimiter settings, and component names that do not conform to Verilog naming rules may be changed to use Verilog escaped identifiers.

- `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

You can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to “[Command Shortcuts](#)” on page 38 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

Prerequisites:

Before using this command, do the following:

- Simulate with **vsim** -uvmcontrol=all.
- Elaborate the UVM testbench by executing the following command:

run 0

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 the Related Topics for links to the other subcommands.

Syntax

uvm uvmpath <questa_uvm_component_path>

Arguments

- <questa_uvm_component_path>
(required) A hierarchical path to an initialized UVM SystemVerilog class object. You can specify the path in the following ways:

Simulation Path — Specify the Simulation Path, which is the hierarchical path mapping for the UVM object within the sim:/uvm_root context, 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 match the simulator's path delimiter settings, and component names do not conform to Verilog naming rules may be slightly changed to use Verilog escaped identifiers.

You can abbreviate the command syntax to the minimum number of characters required to make the command unique. Refer to “[Command Shortcuts](#)” on page 38 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

Adds the specified objects to a VCD file.

Syntax

```
vcd add [-dumports] [-file <filename>] [[-in] [-out] [-inout] | [-ports]] [-internal] [-l <level>] [-nocell] [-r | -r-optcells] <object_name> ...
```

Description

The command allows you to add the following objects:

- Verilog nets and variables
- 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 execute at the same simulation time. The command adds the objects you specify to the VCD header and records their subsequent value changes in the specified VCD file. By default, the file captures all port driver changes and internal variable changes. You can use the command arguments to filter the output.

Related Verilog tasks: \$dumpvars, \$fdumpvars

Arguments

- **-dumports**
(optional) Specifies port driver changes to add to an extended VCD file. When the [vcd dumpports](#) command cannot specify all port driver changes that will appear in the VCD file, you can use multiple vcd add -dumports commands to specify additional port driver changes.
- **-file <filename>**
(optional) Specifies the name of the VCD file. Use this option only when you have used the [vcd files](#) command to create multiple VCD files.

 <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.
- **-l <level>**
(optional) Specifies the number of hierarchical levels to be included in the logging process.
- **-nocell**
(optional) Suppresses the logging of signals within a cell.
- **-r | -r -optcells**
(optional) Specifies that signal and port selection occurs recursively into subregions. Omitting this option limits included signals and ports to the current region. Using **-r** with **-optcells** 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.
- **<object_name> ...**
(required) Specifies the Verilog or VHDL object or objects to add to the VCD file. You can specify multiple objects by separating names with spaces. Accepts wildcards. Must be the final argument to the vcd add command.

Related Topics

[vcd checkpoint](#)

[vcd comment](#)

[vcd dumpsupports](#)

[vcd dumpsupportsall](#)

[vcd dumpsupportsflush](#)

[vcd dumpportslimit](#)

[vcd dumpportsoff](#)

[vcd dumpportson](#)

[vcd file](#)

[vcd files](#)

[vcd flush](#)

[vcd limit](#)

[vcd off](#)

[vcd on](#)

[vcd2wlf](#)

[DumpportsCollapse \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd checkpoint

Dumps the current values of all VCD variables to the specified VCD file.

Syntax

`vcd checkpoint [<filename>]`

Arguments

- <filename>

(optional) Specifies the name of the VCD file. If you omit <filename>, the command executes on the file designated by either the [vcd file](#) command or *dump.vcd* if vcd file was not invoked.

Description

While simulating, dumps only VCD variable value changes. Related Verilog tasks are \$dumpall and \$fdumpall.

Related Topics

[vcd add](#)
[vcd comment](#)
[vcd dumports](#)
[vcd dumportsall](#)
[vcd dumportsflush](#)
[vcd dumportslimit](#)
[vcd dumpportsoff](#)
[vcd dumpportson](#)
[vcd file](#)
[vcd files](#)
[vcd flush](#)
[vcd limit](#)
[vcd off](#)
[vcd on](#)
[vcd2wlf](#)

[DumportsCollapse \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd comment

Inserts the specified comment in the specified VCD file.

Syntax

`vcd comment <comment string> [<filename>]`

Arguments

- `<comment string>`
(required) Comment to include in the VCD file. You must enclose the comment in double quotation marks or curly braces. Must be the first argument to the vcd comment command.
- `<filename>`
(optional) Specifies the name of the VCD file. If you omit `<filename>`, the command executes either on the file designated by the [vcd file](#) command, or on *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 \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd dumports

Creates a VCD file that includes port driver data.

Syntax

```
vcd dumports [-compress] [-direction] [-file <filename>] [-force_direction] [-in] [-out] [-inout]  
[-no_strength_range] [-unique] [-vcdstim] <object_name> ...
```

Description

By default all port driver changes are captured in the file. You can use the arguments detailed below to filter the output. Related Verilog task: \$dumports.

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. This does not create an eVCD file.
- **-file <filename>**
(optional) Creates a VCD file. Defaults to the current working directory and the filename *dumports.vcd*. You can open multiple filenames during a single simulation.
 <filename> — Specifies a filename. Specify with a .gz extension to compress the file.
- **-force_direction**
(optional) Causes vcd dumports 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](#) in the User's Manual for additional information.

- **-unique**
(optional) Generates unique VCD variable names for ports, even if those ports connect 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](#) in the User's Manual for further information.
- **<object_name> ...**
(required) Specifies one or more HDL objects to add to the VCD file. Separate names with spaces to specify multiple objects. Accepts wildcards. Must be 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](#) in the User's Manual 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 resimulates the design using the VCD files in place of the instance source files. Refer to [Replacing Instances with Output Values from a VCD File](#) in the User's Manual 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 \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd dumpportsall

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.

Note

 The related Verilog task is \$dumpportsall.

Syntax

vcd dumpportsall [<filename>]

Arguments

- <filename>
(optional) Specifies the name of the VCD file. If you omit <filename>, the command executes 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 \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd dumpportsflush

Flushes the contents of the VCD file buffer to the specified VCD file.

Note

 The related Verilog task is \$dumpportsflush.

Syntax

`vcd dumpportsflush [<filename>]`

Arguments

- <filename>

(optional) Specifies the name of the VCD file. If you omit <filename>, the command executes 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 \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd dumpportslimit

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.

Syntax

```
vcd dumpportslimit <dumplimit> [<filename>]
```

Description

Related Verilog task: \$dumpportslimit

Arguments

- <dumplimit>
(required) Specifies the maximum VCD file size in bytes. Must be the first argument to the vcd dumpportslimit command.
- <filename>
(optional) Specifies the name of the VCD file. If you omit <filename>, the command executes 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 \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd dumpportoff

Turns off VCD dumping and records all dumped port values as x.

Syntax

`vcd dumpportoff [<filename>]`

Description

Related Verilog task: \$dumpportoff

Arguments

- `<filename>`

(optional) Specifies the name of the VCD file. If you omit `<filename>`, the command executes 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 \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd dumpportson

Turns on VCD dumping and records the current values of all selected ports. Typically used to resume dumping after invoking vcd dumpportsoff.

Note

 The related Verilog task is \$dumpportson.

Syntax

vcd dumpportson [<filename>]

Arguments

- <filename>
(optional) Specifies the name of the VCD file. If you omit <filename>, the command executes 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 \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd file

Specifies the filename and state mapping for the VCD file created by a vcd add command.

Note

 The related Verilog task is \$dumpfile.

Syntax

```
vcd file [-dumports] [-direction] [<filename>] [-map <mapping pairs>] [-no_strength_range]
[-nomap] [-unique]
```

Description

The vcd file command is optional. If you use it, you must issue it before any vcd add commands.

Arguments

- **-dumports**
(optional) Captures 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. Does not create an eVCD file.
- **<filename>**
(optional) Specifies the name of the created VCD file. 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 is the character you want to record 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](#) in the User's Manual for additional information.
- **-nomap**
(optional) Affects only VHDL signals of type std_logic. Specifies to use the std_logic enumeration characters of UX01ZWLH- for values recorded in the VCD file. 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 connect 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 \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd files

Specifies filenames and state mapping for VCD files created by the vcd add command.

Note



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 do not use the -compress argument.
- **-direction**
(optional) Includes driver direction data in the VCD file. Does not create an eVCD 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 is the character you wish to record 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](#)” in the User’s Manual for additional information.
- **-nomap**
(optional) Affects only VHDL signals of type std_logic. Specifies to use the std_logic enumeration characters of UX01ZWLH- for values recorded in the VCD file. 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 connect to the same collapsed net.

Description

The vcd files command is optional. If you use the command, you must issue it before any vcd add commands.

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 dumports](#)
[vcd dumportsall](#)
[vcd dumportsflush](#)
[vcd dumportslimit](#)
[vcd dumpportsoff](#)
[vcd dumpportson](#)
[vcd file](#)

[vcd flush](#)

[vcd limit](#)

[vcd off](#)

[vcd on](#)

[vcd2wlf](#)

[DumpportsCollapse \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd flush

Flushes the contents of the VCD file buffer to the specified VCD file. Related Verilog tasks:
\$dumpflush, \$fdumpflush

Note

 The related Verilog tasks are \$dumpflush and \$fdumpflush.

Syntax

vcd flush [<filename>]

Arguments

- <filename>
(optional) Specifies the name of the VCD file. If you omit <filename>, the command executes on the file designated by the [vcd file](#) command, or on *dump.vcd* if vcd file was not invoked.

Description

Use this command when you want to create a complete VCD file without ending your current simulation.

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 \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd limit

Specifies the maximum size of a VCD file (by default, limited to available disk space).

Note

 The related Verilog tasks are \$dumplimit and \$fdumplimit.

Syntax

`vcd limit <filesize> [<filename>]`

Description

When the size of the file exceeds the limit, a comment is appended to the file and VCD dumping is disabled.

Arguments

- <filesize>

(Required) Specifies the maximum VCD file size, in bytes. The numerical value of <filesize> must be a whole number. Must be 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 you omit <filename>, the command executes on the file designated by the [vcd file](#) command, or on *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 dumports](#)
[vcd dumportsall](#)
[vcd dumportsflush](#)
[vcd dumportslimit](#)

[vcd dumpportoff](#)

[vcd dumpportson](#)

[vcd file](#)

[vcd files](#)

[vcd flush](#)

[vcd off](#)

[vcd on](#)

[vcd2wlf](#)

[DumpportsCollapse \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd off

Turns off VCD dumping to the specified file and records all VCD variable values as x.

Note

 The related Verilog tasks are \$dumpoff and \$fdumpoff.

Syntax

`vcd off [<filename>]`

Arguments

- <filename>

(optional) Specifies the name of the VCD file. If you omit <filename>, the command executes on the file designated by the [vcd file](#) command, or on *dump.vcd* if vcd file was not invoked.

Related Topics

[vcd add](#)
[vcd checkpoint](#)
[vcd comment](#)
[vcd dumports](#)
[vcd dumportsall](#)
[vcd dumportsflush](#)
[vcd dumportslimit](#)
[vcd dumpportsoff](#)
[vcd dumpportson](#)
[vcd file](#)
[vcd files](#)
[vcd flush](#)
[vcd limit](#)
[vcd on](#)
[vcd2wlf](#)

[DumportsCollapse \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcd on

Turns on VCD dumping to the specified file and records the current values of all VCD variables.

Syntax

`vcd on [<filename>]`

Description

By default, vcd on runs automatically at the end of any simulation time invoked by the [vcd add](#) command.

Related Verilog tasks: \$dumpon, \$fdumpon

Arguments

- `<filename>`
(optional) Specifies the name of the VCD file. If you omit <filename>, the command executes on the file designated by the [vcd file](#) command, or on *dump.vcd* if vcd file was not invoked.

Related Topics

[vcd add](#)
[vcd checkpoint](#)
[vcd comment](#)
[vcd dumports](#)
[vcd dumportsall](#)
[vcd dumportsflush](#)
[vcd dumportslimit](#)
[vcd dumpportsoff](#)
[vcd dumpportson](#)
[vcd file](#)
[vcd files](#)
[vcd flush](#)
[vcd limit](#)
[vcd off](#)
[vcd2wlf](#)

[DumportsCollapse \[Questa SIM User's Manual\]](#)

Value Change Dump (VCD) Files [Questa SIM User's Manual]

vcd2wlf

Translates a VCD (Value Change Dump) file into a WLF file that you can display in Questa SIM using the vsim -view argument. Works only on VCD files containing positive time values.

Syntax

```
vcd2wlf [-splitio] [-splitio_in_ext <extension>] [-splitio_out_ext <extension>] [-nocase]
{<vcd filename> | -} <wlf filename>
```

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. To ensure that bits get converted together as a bus, 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

- **-splitio**
(optional) Specifies to split extended VCD port values into their corresponding input and output components, creating two signals instead of just one in the resulting .wlf file. By default, the new input-component signal keeps the name of the original port, while the output-component signal has the original name with an appended "__o".
- **-splitio_in_ext <extension>**
(optional) Adds an extension to input-component signal names created with -splitio.
 <extension> — Specifies a string.
- **-splitio_out_ext <extension>**
(optional) Adds an extension to output-component signal names created with -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 \[Questa SIM User's Manual\]](#)

[Value Change Dump \(VCD\) Files \[Questa SIM User's Manual\]](#)

vcom

Compiles VHDL source code into a specified working library (or to the work library by default).

Syntax

```
vcom [options] <filename> [<filename> ...]

[options]:
[-32 | -64] [-87 | -93 | -2002 | -2008]
[+acc[=<spec>]]] [-addpragmaprefix <prefix>] [-allowProtectedBeforeBody] [-amsstd | -
noamsstd]
[-bindAtCompile] [-bindAtLoad]
[-check_synthesis] [-constimmedassert | -noconstimmedassert] [+cover[=<spec>]] [- -
coverdeglitch {"<n> <time_unit>"}] [-coverenhanced] [-coverexcludedefault] [-coverfec]
[-coveropt <opt_level>] [-coverreportcancelled]
[-debugVA] [-defercheck] [-deferSubpgmCheck | -noDeferSubpgmCheck] [-dirpath
<pathname>]
[-error <msg_number>[,<msg_number>,...]] [-explicit] [-extendedtogglemode 1|2|3]
[(-F | -file | -f) <filename>] [-fatal <msg_number>[,<msg_number>,...]] [-feceffort {1 | 2 | 3}]
[-force_refresh <primary> [<secondary>]] [-fsmdebug] [-fsmimplicittrans] [-fsmmultitrans]
[-fsmsamestatetrans] [-fsmsingle] [-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][pc]] [-modelsimini <path/modelsim.ini>] [-msglimit {error | warning | -
all[,-<msgNumber>,...] | none[,+<msgNumber>,...] | <msgNumber>,[msgNumber,...]}] [- -
msglimitcount <limit_value> -msglimit {error | warning | all[,-<msgNumber>,...] | -
none[,+<msgNumber>,...] | <msgNumber>,[msgNumber,...]}]
[-no1164] [-noaccel <package_name>] [-nocasestaticerror] [-nocheck] [-nocoverclkoptbuiltins]
[-nocoverrespecthandl] [-nocoversub] [-nocreatelib] [-nodbgsym] [-nofrangecheck] [- -
nofsmresettrans] [-noFunctionInline] [-noindexcheck] [-nologo] [-nonstddriverinit] [- -
noothersstaticerror] [-note <msg_number> [<msg_number>, ...]] [-novital] [-novitalcheck]
[-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 abcepx] [-skipsynthoffregion] [-smartdbgsym] [-source] [-stats [=+ | -]<feature>[,[+ | -]<mode>]] [-suppress {<msgNumber> | <msgGroup>} [,<msg_number> | <msgGroup>] ,...]]
 [-togglecountlimit <int>] [-togglewidthlimit <int>] [-toggleportsonly]
 [-version] [-vitalmemorycheck] [-vmake]
 [-warning <msg_number>[,<msg_number>,...]] [-warning error] [-work <library_name>]

Description

You can invoke vcom from Questa SIM or from the command prompt of your operating system. You can invoke this command during simulation.

Compiled libraries change with major versions of Questa SIM. When moving between major versions, you must use the vcom -refresh argument to refresh compiled libraries. You do not need to refresh libraries for minor versions (letter releases).

All arguments to the vcom command are case-sensitive. For example, -WORK and -work are not equivalent.

You can use the -help or -h switch for additional information on the command.

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. These options are ignored if you run vcom from an <install_dir>/<platform>/ directory.

You can specify these options only on the command line; 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](#)” in the User’s Manual 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>]`

(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.

Note



Use of the `+acc` argument with vcom may reduce simulation speed.

`<spec>` — A string of one or more of the following characters:

`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.

- `-addpragmaprefix <prefix>`

(optional) Enables recognition of pragmas with a user specified prefix. If you do not specify a prefix, pragmas are treated as comments.

All regular synthesis and coverage pragmas are honored.

`<prefix>` — Specifies a user defined string. The default is no string, indicated by quotation marks.

You can also set the prefix with the `AddPragmaPrefix` variable in the vcom section of the `modelsim.ini` file. Refer to [AddPragmaPrefix](#) in the User's Manual for more information.

- `-allowProtectedBeforeBody`

(optional) Allows creation of a variable of a protected type 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. Refer to [AmsStandard](#) and [MGC_AMS_HOME](#) in the User's Manual for more information.

- `-assertdebug`

(optional) Allows you to debug SVA/PSL objects when used with `vsim -assertdebug`.

- `-bindAtCompile`

(optional) Forces Questa SIM to perform default binding at compile time rather than at load time. You can change the permanent default by editing the `BindAtCompile` variable in the

modelsim.ini. Refer to “[Default Binding](#)” and [BindAtCompile](#) in the User’s Manual for more information.

- **-bindAtLoad**

(optional) Forces Questa SIM to perform default binding at load time rather than at compile time. (Default)

- **-check_synth**

(optional) Turns on limited synthesis rule compliance checking, checking 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. Refer to [CheckSynthesis](#) in the User’s Manual for more information.

- **-constimedassert**

(optional) Displays immediate assertions with constant expressions in the GUI, in reports, and in the UCDB. By default, immediate assertions with constant expressions display in the GUI, in reports, and in the UCDB. Use this switch only if you have used the **-noconstimedassert** switch previously, or if the ShowConstantImmediateAsserts variable in the vcom section of the *modelsim.ini* file is set to 0 (off). Refer to [ShowConstantImmediateAsserts](#) in the User’s Manual for more information.

- **-noconstimedassert**

(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 can also set the ShowConstantImmediateAsserts variable in the vcom section of the *modelsim.ini* file to 0 (off). Refer to [ShowConstantImmediateAsserts](#) in the User’s Manual for more information.

- **+cover[=<spec>]**

(optional) Enables coverage statistics collection on all design units compiled in the current compiler run. Consider using the **+cover** or **+nocover** arguments to [vopt](#) instead, to specify the precise design units and regions to be instrumented (or uninstrumented) for coverage.

In cases where multiple, potentially competing **+cover** arguments are applied, the coverage option specified later in the command line overrides an option specified before.

The **+cover** argument with no “**=<spec>**” designation is equivalent to “**+cover=bcesft**”.

<spec> — A string of one or more of the following characters, with no spaces:

b — Collect branch statistics.

c — Collect condition statistics. Collects only FEC statistics.

e — Collect expression statistics. Collects only FEC statistics.

s — Collect statement statistics.

t — Collect toggle statistics. Overridden if ‘**x**’ is specified elsewhere.

x — Collect extended toggle statistics (Refer to “[Toggle Coverage](#)” in the User’s Manual 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.

- **-coverenhanced**

(optional) Enables non-critical functionality, which can change the appearance or content of coverage metrics. This argument has an effect only in letter releases (10.0a, 10.0b, and so on). Major releases (10.0, 10.1, and so on) enable all coverage enhancements present in previous letter release streams by default, so **-coverenhanced** is not necessary. Bug fixes important to the accuracy of coverage numbers are also always enabled by default. Because the effect of **-coverenhanced** varies from release to release, the details of the enhancements it enables are in the product release notes, rather than in the Command Reference. To view these details, search the release notes for "coverenhanced".

- **-coverexcludedefault**

(optional) Excludes VHDL code coverage data collection for the OTHERS branch in both Case statements and Selected Signal Assignment statements. Excludes all forms of code coverage collection for statements contained in the OTHERS branch.

- **-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. Refer to [CoverFEC](#) in the User's Manual.

- **-nocoverclkoptbuiltins**

(optional) 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 they were excluded because of the "clock optimization". You can turn off optimization with **-nocoverclkoptbuiltins**. You can customize the default behavior with the CoverClkOptBuiltins variable in the *modelsim.ini* file. For more information, refer to [CoverClkOptBuiltins](#) and "[Missing Branches in VHDL and Clock Optimizations](#)" in the User's Manual.

- **-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.

You must enclose **<n> <time_unit>** in quotation marks (“ ”) when you put a space between **<n>** and **<time_unit>** (for example, 2ps or “2 ps”).

If a process or a continuous assignment is evaluated more than once during any period of length “**<n> <time_unit>**”, only the final execution during that period is added to the

coverage data for that process/continuous assignment. If you specify a deglitch period of zero, 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. Refer to [CoverDeglitchOn](#) and [CoverDeglitchPeriod](#) in the User's Manual.

- **-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 can 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.

The default optimization level is 3. You can edit the CoverOpt variable in the *modelsim.ini* file to change the default. Refer to [CoverOpt](#) in the User's Manual.

- **-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 enable this behavior with the CoverReportCancelled *modelsim.ini* variable. Refer to [CoverReportCancelled](#) in the User's Manual.

- **-createlib[=compress]**

Creates libraries that do not currently exist.

compress — Compresses the created libraries

- **-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 it encounters them in 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.
- **-dirpath <pathname>**
(optional) Specifies the location of a working directory to store in the library to override the current working directory. This allows you hide the directory path information.

Caution

Use of this argument is not recommended.

For example, if you use **-dirpath** to override the working directory information, when an end-user selects something in the design and asks to see the declaration, the Questa SIM user interface will not be able to find the source files.

- **-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 [error](#) and "[Message Severity Level](#)" in the User's Manual for more information.

 <msg_number> — A number identifying the message. Specify multiple message numbers 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. This behavior does not match the VHDL standard, but the majority of EDA tools choose explicit operators over implicit operators, so activation of 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. Refer to [ExtendedToggleMode](#) in the User's Manual.

- (-F | -file | -f) <filename>

(optional) -f, -file and -F each specifies an argument file with more command-line arguments, allowing you to use complex argument strings without retyping. You can nest -F, -f and -file commands. 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 “[Argument Files](#)” on page 37 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 [fatal](#) and “[Message Severity Level](#)” in the User’s Manual for more information.

<msg_number> — A number identifying the message. Specify multiple message numbers as a comma-separated list.

- -feceffort {1 | 2 | 3}

(optional) The recommended option to set the level of effort for FEC. 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 [FecEffort](#) 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. Updates the work directory by default; 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 refresh.

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**

(optional) Allows access to finite state machines for debugging.

- **-fsmimplicittrans**

(optional) Toggles recognition of implied same state transitions, which is off by default.

Multiple command arguments are available for FSM management. You can use any combination of the FSM arguments. Refer to [Advanced Command Arguments for FSMs](#) in the User's Manual.

- **-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.

Multiple command arguments are available for FSM management. You can use any combination of the FSM arguments. See [Advanced Command Arguments for FSMs](#) in the User's Manual.

- **-nofsmresettrans**

(optional) Toggles recognition of synchronous or asynchronous reset transitions, and is on by default.

This includes/excludes reset transitions in coverage results.

Multiple command arguments are available for FSM management. You can use any combination of the FSM arguments. See [Advanced Command Arguments for FSMs](#) in the User's Manual.

- **-fsmsamestatetrans**

(optional) Specifies to recognize transitions to the same state as valid. By default, the compiler would not consider S0->S0 as a valid transition in the following example:

```
If(cst == S0)
    nst = S0
```

- **-fsmsingle**

(optional) Toggles the recognition of VHDL FSMs where the current state variable is of type std_logic, bit, boolean, or single-bit std_logic_vector/bit_vector and Verilog single-bit FSMs.

Multiple command arguments are available for FSM management. You can use any combination of the FSM arguments. See [Advanced Command Arguments for FSMs](#) in the User's Manual.

- **-fsmverbose [b | t | w]**

(optional) Provides information about FSMs detected, including state reachability analysis. You can specify any or none of the following values:

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.

If you do not specify a value, this argument 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. Does not allow wildcards and multiple design unit names.

<**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 you do not fully specify an explicit binding, defaults for the architecture, port maps, and generic maps are used as needed. Edit the `RequireConfigForAllDefaultBinding` *modelsim.ini* variable to set a permanent default. Refer to [RequireConfigForAllDefaultBinding](#) and [Default Binding](#) in the User's Manual for more information.

- **-ignorepragmaprefix <prefix>**

(optional) Directs vcom to ignore pragmas with the specified prefixname. All affected pragmas are treated as regular comments. Edit the `IgnorePragmaPrefix` *modelsim.ini* variable to set a permanent default. Refer to [IgnorePragmaPrefix](#) in the User's Manual.

<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. Refer to [ignoreStandardRealVector](#) in the User's Manual. 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, the assumption is 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 `InitOutCompositeParam` variable to 1 in the *modelsim.ini* file. Refer to [InitOutCompositeParam](#) in the User's Manual.

- **-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, designs 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 InitOutCompositeParam variable to 2 in the *modelsim.ini* file. Refer to [InitOutCompositeParam](#) in the User's Manual.

- **-just abcepx**

(optional) Directs the compiler to include only the following:

- a — architectures
- b — bodies
- c — configurations
- e — entities
- p — packages
- x — VHDL 2008 context declarations

You can use any combination in any order, 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 TranscriptFile or BatchTranscriptFile *modelsim.ini* variables. (Refer to [TranscriptFile](#) or [BatchTranscriptFile](#) in the User's Manual.) 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> —

- **-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 applies only to designs compiled using -87. If you compile using -93, such cases are flagged as a warning or error, even without the -lint argument.

You can also enable lint checking with the Show_Lint variable in the *modelsim.ini* file. Refer to [Show_Lint](#) in the User's Manual.

- **-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. Refer to [PreserveCase](#) in the User's Manual.

- **-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 to the description of the [oldVHDLConfigurationVisibility](#) *modelsim.ini* variable in the User's Manual, for information on processing visibility of VHDL component configurations consistent with prior releases.

- **-mixedsvvh [b | l | r][i][pc]**

(optional) Facilitates using VHDL packages at the SystemVerilog-VHDL boundary of a mixed-language design. When you compile a VHDL package with -mixedsvvh, you can include the package in a SystemVerilog design as if it were defined in SystemVerilog itself.

If you specify -mixedsvvh without arguments, Questa SIM 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.

pc — Preserves case of all identifiers.

- **-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 name. Use a forward slash (/) as the path separator on Windows systems.

- **-msglimit {error | warning | all[-<msgNumber>,...] | none[+<msgNumber>,...] | <msgNumber>,[msgNumber,...]}**

(optional) Limits messages to the default message limit count of five. Use the -msglimitcount argument to change the default message limit count. Specify multiple messages as a comma-separated list.

error — Stops the run when the total number of errors reaches the default count.

warning — Stops the run when the total number of warnings reaches the default count.

all — Limits all messages to the default count except those you specifically exclude. To exclude messages from the limit, supply a comma-separated list of message numbers, each preceded by a minus sign (-), for example “all,-<msgNumber1>,<msgNumber2>,...”.

none — Excludes all messages from the default count except those you specifically include. To include messages to limit to the default count, supply a comma-separated list of message numbers, each preceded by a plus sign (+), for example “none,+<msgNumber1>,+<msgNumber2>,...”.

<msgNumber>[,<msgNumber>,...] — Specifies messages to limit to the default count.

Examples:

- Stop the run when the total number of errors reaches the default count:

```
-msglimit error
```

- Stop the run when the total number of warnings reaches the default count:

```
-msglimit warning
```

- Specifically limit messages 2374 and 2385 to the default count:

```
-msglimit 2374,2385
```

- Limit all messages to the default count, except messages 2374 and 2385:

```
-msglimit all,-2374,-2385
```

- Limit only messages 2374 and 2385 to the default count:

```
-msglimit none,+2374,+2385
```

- -msglimitcount <limit_value> -msglimit {error | warning | all[-<msgNumber>,...] | none[+<msgNumber>,...] | <msgNumber>,[msgNumber,...]}
- (optional) Specifies to either stop the run when the total number of reported errors/warnings reaches the user-defined limit value, or to limit the reporting of listed messages to the user-defined limit_value. Overrides the MsgLimitCount variable in the *modelsim.ini* file. Refer to [MsgLimitCount](#) in the User's manual.
- -no1164
- (optional) Causes source files to be compiled without taking advantage of the built-in version of the IEEE std_logic_1164 package. This typically results 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 that are globally static are allowed. This switch prevents the compiler from warning on such expressions. If you specify the **-pedanticerrors** switch, this switch is ignored.
- **-nocheck**
(optional) Disables index and range checks. You can disable these individually using the **-noindexcheck** and **-norangecheck** arguments, respectively.
- **-nocoverrespecthandl**
(optional) Specifies to automatically convert the VHDL 'H' and 'L' input values on conditions and expressions to '1' and '0', respectively. By default, they are not automatically converted.
As an alternative to this argument — if you are not using 'H' and 'L' values — you can do either of the following:
 - Change your VHDL expressions of the form (a = '1') to (to_x01(a) = '1') or to **std_match(a,'1')**.
 - Set the **CoverRespectHandL** variable in the *modelsim.ini* file to 0. Refer to [CoverRespectHandL](#) in the User's Manual.
- **-nocovershort**
(optional) Disables short circuiting of expressions when coverage is enabled. Short circuiting is enabled by default. You can use the **CoverShortCircuit** variable in the *modelsim.ini* file to customize the default behavior. Refer to [CoverShortCircuit](#) in the User's Manual.
- **-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. Refer to [CoverSub](#) in the User's Manual.
- **-nocreatelib**
(optional) Stops automatic creation of missing work libraries. Overrides the **CreateLib** *modelsim.ini* variable. Refer to [CreateLib](#) in the User's Manual.
- **-nodebug[=ports]**
(optional) Hides the internal data of all compiled design units within the GUI and other parts of the tool.
 - nodebug — When specified with no options:
 - Does not hide ports, leaving port information available for instantiation in a parent scope.

- The design units' source code, internal structure, registers, nets, and so on, do not display in the GUI.
- You cannot access the hidden objects through the Dataflow or Schematic window or with commands.
- You cannot set breakpoints or single step within hidden code. Do not compile with this switch until you are done debugging.

-nodebug=ports — Also hides the ports for the lower levels of your design; best used to compile only 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. To encrypt regions within the file, use the `protect compiler directive.

Design units or modules compiled with -nodebug can only instantiate design units or modules that are also compiled with -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 are source window annotation and textual dataflow.

Specify this switch only if no one using the library will require this information for design analysis purposes.

- **-noDefSubpgmCheck**

(optional) Causes range and length violations detected within subprograms to be reported as errors (instead of as warnings). An alternative to using this argument is to set the NoDefSubpgmCheck variable in the *modelsim.ini* file to a value of 1. Refer to [NoDefSubpgmCheck](#) in the User's Manual.

- **-nofprangecheck**

(optional) Disables range checks on floating type values only.

- **-noFunctionInline**

(optional) Turns off VHDL subprogram inlining for design units that use a local copy of a VHDL package. Useful when 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 did have explicit initial values. This could cause Questa SIM to incorrectly use the actual signal's initial value when initializing lower level drivers. This argument does not cause all lower-level drivers to use the actual signal's initial value, only the specific cases where older versions used the actual signal's initial value.
- **-noothersstaticerror**
(optional) Disables warnings caused by array aggregates with multiple choices having "others" clauses that are not locally static. If you specify **-pedanticerrors**, 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](#)" in the User's Manual 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.
- **-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 [note](#) and "[Message Severity Level](#)" in the User's Manual for more information.

 `<msg_number>` — A number identifying the message. Specify multiple message numbers 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. Enables you to set breakpoints in the VITAL behavior process, and permits single stepping through the VITAL procedures to debug your model. Also, enables viewing of all of the VITAL data 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).
- **-nowarn <category_number>**
(optional) Selectively disables a category of warning messages. You can disable all warnings for all compiles through the Main window **Compile > Compile Options** menu command, or the *modelsim.ini* file (refer to [modelsim.ini Variables](#) in the User's Manual).
 <category_number> — Specifies one or more numbers corresponding to the categories in [Table 3-8](#). Specify multiple message categories as a comma-separated list.

Table 3-8. Warning Message Categories for vcom -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 (“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 cannot handle
14	locally static error deferred until simulation run

- **-O0 | -O1 | -O4 | -O5**
(optional) Controls optimization aggressiveness. Refer to ["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 can cause event order differences in your simulation and can negatively impact performance. Useful for analyzing the behavior of the simulator.
- O4 — (default) Enable standard SE optimizations. 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. Attempts to optimize loops, and prevents variable assignments in situations where a variable is assigned but not actually used.

- **-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 to the description of the [OldVHDLConfigurationVisibility](#) *modelsim.ini* variable in the User's Manual, for more information.

- **-optionset <optionset_name>**

(optional) Calls an optionset as defined in the *modelsim.ini* file. Refer to the section “[Optionsets](#)” on page 35 for more information.

- **-outf <filename>**

(optional) Specifies a file to save the final list of options to, 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](#)” in the User's Manual for a complete list of these conditions. This argument overrides -nocasestaticerror and -noothersstaticerror.

You can view a complete list of errors by executing the command:

```
verror -kind vcom -pedanticerrors
```

- **-performdefaultbinding**

(optional) Enables default binding that has been disabled through the [RequireConfigForAllDefaultBinding](#) option in the *modelsim.ini* file. Refer to [RequireConfigForAllDefaultBinding](#) in the User's Manual.

- **-preserve**

(optional) Forces vcom to preserve the case of letters in object identifiers. You can also preserve letter case with the [PreserveCase](#) variable in the *modelsim.ini* file. Refer to [PreserveCase](#) in the User's Manual.

- **-[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>. Customer Support uses output from these arguments 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 in 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.

The command line +protect argument will be fully deprecated in a future release. The simulator commands will issue a warning message any time you use the +protect argument.

<filename> — A user specified string that specifies the name for the encrypted output file. Creates an encrypted output file in the current directory.

- **-pslfile <filename>**

(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](#)” in the User’s Manual for more information.

<filename> — An existing PSL assertion file.

- **-quiet**

(optional) Disables output of informative messages about processing the design, such as Loading, Compiling, or Optimizing, but not messages about the design itself.

- **-rangecheck**

(default) Enables run time range checking. You can disable range checking with the -norangecheck argument.

- **-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 delete.

This option is not supported for SystemC modules.

- If <primary> is an entity — Refreshes only that entity, no related architectures.
- If <primary> is a package — Refreshes only the package, and the only legal value of <secondary> is “body”.

- If you specify both <primary> and <secondary> — Updates only the <secondary> architecture, 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 can use a specific design name with -refresh to regenerate a library image for that design, but you cannot 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. Use this argument only 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.
- -showsubprograms | -noshowsubprograms
(optional) Toggles viewing of VHDL subprogram scopes between the command line and a GUI window, for example, the Structure window. The default is not to show subprogram scopes.
- -skip abcepX
(optional) Directs the compiler to skip all:
 - a — architectures
 - b — bodies
 - c — configurations
 - e — entities
 - p — packages
 - x — VHDL 2008 context declarations

You can use any combination in any order, but the argument requires at least one option.

- -skipsynthoffregion
(optional) Ignore all constructs within synthesis_off or translate_off pragma regions.
- -smartdbgsym
(optional) Reduces the size of design libraries by minimizing the number 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. Refer to [SmartDbgSym](#) in the User's Manual.

- **-source**

(optional) Displays the associated line of source code before each error message generated during compilation. By default, only the error message displays.

- **-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).

Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes effect.

[+ | -] — Controls activation of the feature or mode. The plus character (+) enables the feature, and the minus character (-) disables the feature. You can also enable a feature or mode by specifying it without the plus (+) character. Setting this switch adds or subtracts features and modes from the settings in the Stats *modelsim.ini* variable.

<feature>

all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with the none option. Applied first when specified in a string with other options.

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 the all option. Applied first when specified in a string with other options.

perf — Display time and memory performance statistics.

time — (default) Display Start, End, and Elapsed times.

Note



Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

Modes can be set for a specific feature, or globally for all features. Use the plus (+) or minus (-) character to add or subtract a mode for a specific 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 kilobyte units with no auto-scaling.

list — Display statistics in a Tcl list format when available.

verbose — Display verbose statistics information when available.

Note

Specify vcom -quiet to disable all -stats features.

- **-suppress {<msgNumber> | <msgGroup>} [,<msg_number> | <msgGroup>] ,...]**
 (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message 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 [suppress](#) and “[Message Severity Level](#)” in the User’s Manual for more information.

<msgNumber>

A specific message number

<msgGroup>

A value identifying a pre-defined group of messages, based on area of functionality. These groups only suppress notes or warnings. The valid arguments are:

All, GroupNote, GroupWarning, GroupFLI, GroupPLI, GroupSDF, GroupVCD, GroupVital, GroupWLF, GroupTCHK, GroupPA, GroupLRM

- **-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. Refer to [ToggleCountLimit](#) in the User’s Manual.

<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. (To see coverage of all ports in the design, use the “vopt +acc=p” command — but note that this command turns off inlining, and could cause a significant performance penalty.) You can selectively enable toggle coverage on specific ports with the “vopt +acc=p+<selection>” command. Overrides any global value set by the TogglePortsOnly *modelsim.ini* variable. Refer to [TogglePortsOnly](#) in the User’s Manual.

- **-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. Refer to [ToggleWidthLimit](#) in the User’s Manual.

<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. The **vmake** command uses this data 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 or not they contribute data to the initial compile. This switch can increase compile time in addition to increasing the accuracy of the compile. Refer to 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 **warning** and "**Message Severity Level**" in the User's Manual for more information.
 <msg_number> — A number identifying the message. Specify multiple message numbers 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, 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 when you specify -refresh or -force_refresh) Specifies the name of the file containing the VHDL source to compile. Requires at least one filename. You can specify multiple filenames as a space-separated list. You can use wildcards, for example, *.vhd.
If you do not specify a filename, and you are using the GUI, a pop-up dialog box enables you to select 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; subsequent users 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, meaning that, 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. VHDL users typically expect that the explicit declaration can to hide the implicit declaration.

```
ARITHMETIC. "=" (left, right)
```

- The -work argument specifies mylib as the library to regenerate. The -refresh argument 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, and a message count summary. Disables echoing of the command line.

```
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

Used to set the display of attributes of a specified coverage database file or coverstore during batch mode simulation.

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]
```

Description

This command displays the following types of attributes:

- Test Attributes — Sets the value of attributes for testcase information. Refer to "[Predefined Attribute Data](#)" in the User's Manual for a complete list of these attributes.

Arguments

- **-32 | -64**

(optional) Specifies whether vcover attribute uses the 32- or 64-bit executable. The default is 32-bit. These options apply only to the supported Linux operating systems.

- **-concise**

(optional) Prints attribute values only, without the attribute names, for the resulting query. In some circumstances, this setting can shorten processing time.

- **-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. Specify multiple message numbers as a comma-separated list.

Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to [error](#) and "[Message Severity Level](#)" in the User's Manual 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 display.

- <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 name. On Windows systems, use a forward slash (/) as the path separator.
 - -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. Specify multiple attributes as a space-separated list. Refer to [Predefined Attribute Data](#) in the User's Manual 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).

Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Set this switch to add or subtract features and modes from the default settings "cmd,msg,time".
- <feature>*
- 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.

Note



Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

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 kilobyte 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. Specify multiple message numbers as a comma-separated list.

Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to [note](#) and [“Message Severity Level”](#) in the User’s Manual 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>. Customer Support uses output from these arguments 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 to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

<msg_number> — A number identifying a specific message. Specify multiple message numbers as a comma separated list.

Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to [“suppress”](#) in the User’s Manual.

- **-tcl**
(optional) Prints attribute information in a Tcl list format. 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. Does not support wildcards. 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. Specify multiple message numbers as a comma-separated list.

Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to [warning](#) and “[Message Severity Level](#)” in the User’s Manual 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 \[Questa Verification Management User's Manual\]](#)

[Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)

[vcover merge](#)

[vcover ranktest](#)

[vcover stats](#)

vcover diff

Reports the coverage differences between two UCDBs.

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/scoped>] [-stats [=+ | -]<feature>,[,+ | -]<mode>]] [-structural] [-tdb <name>] [-testname <name>] <UCDB1> <UCDB2> [-verbose]
```

Description

This command can write the results of the coverage differences into a trend database (.tdb) and open it for viewing in the Results Analysis window. You can filter the output so it doesn't show all scopes in the design (which may be very large).

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.

You can use the -gui switch to bring up the results in the GUI. For information on viewing the results in the GUI, refer to “[Viewing vcover diff Results in the Results Analysis Window](#)” in the Verification Management User’s Manual.

For information on interpreting textual output of the diff, including the BNF syntax, refer to the UCDB API Reference Manual.

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 you specify a single character, it is used as both opening and closing delimiter. If you specify two characters, 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 send results to stdout.
- **-gui**
(optional) Open the Results Analysis window in the GUI and populate with the specified .tdb. The default file read in is QuestaDiff.tdb.
- **-flagfield**
(optional) Displays flag field as single hex number difference. Default is to display flag bits individually.
- **-memblk**
(optional) Specifies 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>. Customer Support uses output from these arguments 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 in the *\$MTI_HOME/vm_src/* directory. You can write your own rulesfile, and then specify that file with this command.

- **-scope <path/scope>**

(optional) Filters the data to show only the specified design unit, testplan, ordesign hierarchical scope. You must specify the full hierarchical path in <path/scope>. You cannot use wildcards.

- **-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).

You can specify multiple features and modes for each instance of -stats in a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the default settings "cmd,msg,time".

<feature>

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.

Note

 Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

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 kilobyte 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 created triage database. Default base name matches the name specified with **-diffsfile**. Otherwise, it creates a file called QuestaDiff.tdb.
- **-testname <name>**
(optional) Specifies the name for the testplan column in the GUI. The default name is QuestaDiff.
- **<UCDB1>**
(required) Designates input filename (include .ucdb file extension) to compare against **<UCDB2>**.
- **<UCDB2>**
(required) Designates input filename (include .ucdb file extension) to compare 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

- Report 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
- Report 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 \[Questa SIM User's Manual\]](#)

[Viewing vcover diff Results in the Results Analysis Window \[Questa Verification Management User's Manual\]](#)

[vcover attribute](#)

[vcover dump](#)

[vcover merge](#)

[vcover ranktest](#)

[vcover report](#)

[vcover stats](#)

[vcover testnames](#)

vcover dump

Produces and prints to *stdout* a textual description of the contents of the UCDB file.

Syntax

```
vcover dump <ucdb_file> [-32 | -64] [-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 print 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. Specify multiple message numbers as a comma-separated list.
Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to [error](#) and "[Message Severity Level](#)" in the User's Manual 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. Specify multiple message numbers as a comma-separated list.
Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to [note](#) and "[Message Severity Level](#)" in the User's Manual 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>. Customer Support uses output from these arguments 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).

Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the default settings "cmd,msg,time".

<feature>

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.

Note

 Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

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 kilobyte 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. Specify multiple message numbers as a comma-separated list.

Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to “[suppress](#)” in the User’s Manual for more information.

- -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. Specify multiple message numbers as a comma-separated list.

Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to [warning](#) and “[Message Severity Level](#)” in the User’s Manual 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

Provides details on the creation process of a specific UCDB file.

Syntax

```
vcover history < hist_options > <file>  
< hist_options > = [-32 | -64] [-full] [-display|-nodisplay ( test | merge | testplan )]* [-  
[w]prof=<filename>] [-proftick=<integer>] [-stats [=+ | -]<feature>,[,+ | -]<mode>]]
```

Description

This command tells you whether the UCDB was created by multiple commands over time, through scripts or regression runs, and so on. You can use the output to re-create the UCDB.

On merged/trend UCDB files, the vcover history command outputs a list — in script form — of the commands used to create any and all of the UCDBs that were merged to create the specified (top level) file. By default, the output includes both the xml2ucdb (testplan) and vcover merge (merge) commands required to re-create the merged UCDB. The command includes options to control further output detail.

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, meaning that the output does not include test creation commands.
 - merge - includes/excludes the commands used to create the merged UCDB. By default this option is set to -display merge, meaning that the output includes “vcover merge” creation commands.
 - testplan - includes/excludes the commands used to create the testplan UCDB. By default, this option is set to -display testplan, meaning that the output includes “xml2ucdb” creation commands.
- -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).

Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the default settings "cmd,msg,time".

<feature>

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.

Note

 Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

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>. Customer Support uses output from these arguments 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

Following is an example of how the resulting output appears:

```
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-level 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 tracker.ucdb -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 tracker.ucdb -stats=time,cmd,msg -stats=none,perf

Related Topics

[vcover merge](#)

[vcover ranktest](#)

[vcover stats](#)

vcover merge

The vcover merge command merges UCDB files, saved with the coverage save command, or coverstores, created with the -coverstore option to vsim, into a single UCDB output file.

Syntax

```
vcover merge <merge_options> {[-out] <outfile> <input1> [<input2> ... <inputn>]}
```

<merge_options> =

```
[ -32 | -64] [-and] [-append] [-backup] [-checkinputs] [-date <yyyymmddhhmmss>] [-forceall] [-inputs <cmdfile>] [-install <path>] [ [-instance <path>] [-instance <path>] ... [-recursive]] | [-du <du_fullname>] [-recursive] [-du [-recursive]]... ] [-level <1-3>] [-lockingfull | -lockingtimeout | -lockingnone] [-logfile <filename>] [-l <filename>] [-master <ucdb_filename>] [-mergebytokenno] [-mergedesigndiffs] [-notagging] [-outputstore <output_coverstore_dir>] [-pathseparator <char>] [-preservetestcount] [-[w]prof=<filename>] [-proftick=<integer>] [-quiet] [-silent] [-stats [=+[ + -]<feature>,[ [+ -]<mode>]]] [-strip <n>] [-ucdbchecks] [-verbose] [-version] [-combine | -combinemax | -combinemin | -totals | -testassociated] [-timeout <seconds>] [-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>,...]]]
```

Description

The vcover merge command creates a union of two or more UCDB files, regardless of the hierarchy in each file, or coverstores, however they must have the same hierarchy. Refer to the section [Coverage Auto-save Coverstore](#) for a detailed discussion on merging coverstores.

By default, vcover merge does not store test-specific coverage information. To keep test-specific information (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:

- the [coverage analyze](#) command, or any Test Analysis features in the GUI.
- the [vcover ranktest](#) command, or any Ranking operations in the GUI.

If you want to preserve individual UCDBs for analysis and ranking, test-associated merging is not advised.

The tool issues warning message #6846, indicating that the resulting merged file could not completely represent the merged information, when:

- "at_least" is greater than 1.
- weights are different for the same object in different files.
- there are different objects in different files.

For these situations, use the -verbose argument with the vcover merge command to obtain further details about potential issues with the merge.

Questa SIM generates a warning if a code coverage instance in a UCDB is changed. You can disable warnings with the -quiet option.

You can invoke vcover merge in the Questa SIM GUI or at the system prompt. Use the -help (or -h) argument to view the command syntax.

For more information, refer to the section [Limitations of Merge for Coverage Analysis](#) in the User's Manual.

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 to append progress messages 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.
- **-checkinputs**
(optional) Checks for valid merge inputs. Ignores invalid input files with a warning message and removes those files from the merge list.
- **-combine | -combinemax | -combinemin**
(optional) Merges two or more different runs of a single test, for re-joining stripped versions of a UCDB file. The differences between the three variants of the -combinexxx 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 are saved in the 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 <yyyymmddhhmmss>

(optional) Overrides the timestamp of the merged file. The <yyymddhhmmss> is the timestamp value in the form of year, day, month, hour, minute, and second.

- -du <du_fullname> [-recursive] [-du [-recursive]]...

(optional) Instructs the tool to merge all instances of the specified design unit in all of the input UCDB files. If you specify one -du entry, one instance of the design unit is created, at the top level, containing all of the merged data. If you specify multiple -du arguments, 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_fullname> 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.

Note

 You cannot use the -du <du_fullname> argument with the -instance <path> argument. Use either -du <du_fullname> or -instance <path>.

- -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. Specify multiple message numbers as a comma-separated list.

Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to [error](#) and “[Message Severity Level](#)” in the User’s Manual for more information.

- -forceall

(optional) Forces all toggles and statements of an instance to merge.

- **-inputs <cmdfile>**
(optional) Specifies a text file containing vcover command arguments and the inputs that you want to merge.

 <cmdfile> — The name of the command text file to merge.
- **-instance <path> [-instance <path>] ... [-recursive]**
(optional) Instructs the tool to merge all occurrences of the specified instances in all the input files, resulting in an output file consisting of a single instance, containing all the merged data. Allows multiple -instance <path> specifications.

 <path> — The path for the specified instance(s). You can change the resulting path using the -install <path> option.

 -recursive — Instructs the tool to merge the complete design subtree, from the designated instances down. If not specified, merges only the specified level. The -recursive option applies to all designated instances. You cannot use multiple -recursive specifications.

Note

 You cannot use the -instance <path> argument with the -du <du_fullname> argument. Use either -instance <path> or -du <du_fullname>.

- **-install <path>**
(optional) Adds additional hierarchy on the front end of instance and object names in the data files. This argument enables you to merge coverage results from simulations that have different hierarchies. Refer to “[Merge Usage Scenarios](#)” in the User’s Manual for more information.

 <path> — A user specified path.
- **-level <1-3>**
(optional) Specifies the merge level: totals (level 1), testassociated (level 2), or preservetestcount (level 3)
- **-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 accurate only 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 stdout and are echoed to any explicitly supplied log file.
 <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](#)” in the User’s Manual.
 <ucdb_filename> — Specified master file, which should be the file containing 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.
- **-mergebytokenno**
(optional) Forces the merge to use token number instead of item number for identification.
- **-mergedesigndiffs**
(optional) Instructs the tool to ignore design unit signature checking, and continue merging. Issues a warning message. Do not use this argument without first validating that the differences in code between the merges of two versions of the same input are expected and approved. Refer to “[Merging and Source Code Mismatches](#)” in the User’s Manual for further details on the use of this argument.
- **-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. Specify multiple message numbers as a comma-separated list.

Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to [note](#) and “[Message Severity Level](#)” in the User’s Manual for more information
- **[-out] <outfile>**
(Specification of -out is optional, <outfile> is required). Sends the merged output to a file. By default, uses the base name of the first UCDB file listed in the command as the output file name. When you specify -out, you can include this argument set anywhere on the command line.

<outfile> — A user specified string for the merged output file name.

- **-outputstore <output_coverstore_dir>**
(optional) Specifies an output directory in which to store the raw, merged coverage output, instead of in a self-contained UCDB file. This is useful when 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. You can use wildcards for the selection of tests to merge.
- **-pathseparator <char>**
(optional) Specifies the path separator character to use in the resulting output UCDB file. <char> must be a non-alphanumeric character, including any of the following characters within the quotation marks: “-=~!@%^&+|./:;”.
- **-preservetestcount**
(optional) Merges and keeps the individual counts of each bin. Also called level 3 merge.
- **-[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>.
- **-proftick=<integer>**
(optional) Sets the time interval between the profile data collections. Default = 10.
- **-quiet**
(optional) Disables warnings when merging databases and a changed instance is encountered.
- **-silent**
(optional) Disables printing of a banner or other messages. Allows printing of error and warning messages only.
- **-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).
Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the default settings “cmd,msg,time”.

<feature>

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.

Note



Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

You can set modes 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 kilobyte 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. Enables you to merge coverage results from simulations that have different hierarchies. Refer to "[Merge Usage Scenarios](#)" in the User's Manual 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. Specify multiple message numbers as a comma-separated list.

Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to [suppress](#) in the User's Manual.

- -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 is 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.

- **-ucdbchecks**
(optional) Reports more verbose messages on UCDB file.
- **-verbose**
(optional) Enables summary code coverage statistics to be computed and directed to the log file each time a file is merged into the database. 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. Specify multiple message numbers as a comma-separated list.

Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to [warning](#) and “[Message Severity Level](#)” in the User’s Manual for more information.
- **<input1> [<input2> ... <inputn>]**
(required) Specifies the input to be merged. You can use wildcards and specify multiple inputs in a space-separated list.

<input>

The input can take either of the following forms

<ucdbfile>

UCDB created by a coverage save command.

<coverstore>[:<testname>[,<testname> ...]]

Coverstore created by the -coverstore argument to the vsim command with optional testnames in a comma-separated list.

Examples

- Merge coverage statistics for myfile1 and myfile2 and write 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 myfile.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 \[Questa SIM User's Manual\]](#)

[Merge Usage Scenarios \[Questa SIM User's Manual\]](#)

[Verification Management Overview \[Questa Verification Management User's Manual\]](#)

["Merging Verification Plan with Test Data" \[Questa Verification Management User's Manual\]](#)

[Analyzing Trends \[Questa Verification Management User's Manual\]](#)

[vcover attribute](#)

[vcover ranktest](#)

[vcover stats](#)

vcover parallelmerge

Performs a merge on a UCDB or CoverStore database using parallel processes. Automatically merges any intermediate merge results, and produces final merged results.

Syntax

vcover parallelmerge <options>

<options> =

[-32 | -64] [-covmode <ucdb/covstore>] [-dbname <file_pattern>] [-filelist <filepath>]
[-genlist] [-genlistfrom <pathdir_or_list>...] [-gridopts <options>] [-gridslots <val>]
[-hostlist <pattern>] [-j <val>] [-log <filename>] [-maxrerun <integer>] [-mergererun]
[-mergetype <totals/testassociated/preservetestcounts>] [-mergeopts <options>]
[-mtimeout <val>] [-outname <name>] [-qtimeout <val>] [-quiet] [-rmdb <path>]
[-runmode <local/rsh/lsh/sge/uge/rtda>] [-stats [=+ | -]<feature>[,[+ | -]<mode>]] [-tempdir
<dir_path>]
[-teststatus <all/passed/failed>] [-verbose]

Description

You can invoke this command in the Questa SIM GUI or at the system prompt. It optimizes the number of parallel merge jobs so that each merge job has at least 2 files to merge. For example, if you command 8 files to merge with 8 parallel jobs, vcover parallelmerge automatically reduces the number of parallel merge jobs to 4.

By default, this command returns the following:

- Runtime and memory consumption by each parallel merge process
- Merge-related error and warning messages.

Arguments

- -32 | -64
(optional) Specifies the executable platform.
- -covmode <ucdb/covstore>
(optional) Specifies type of coverage database, UCDB or CoverStore (Default: ucdb).
- -dbname <file_pattern>
(optional) Specifies UCDB or CoverStore file patterns to search in the database location specified with either -genlist or -genlistfrom. (Default: “*.ucdb” for UCDB and “*” for Coverstore).
- -filelist <filepath>
(optional) Specifies file list name and path containing the list of files to merge. If the tool generates the list, it does so in the file name specified by this option (Default: \$cwd/parallelist).

- **-genlist**
(optional) Generates the file list from \$cwd for UCDB mode, or \$cwd/CoverStore for Coverstore mode, in the location specified by the -filelist option. The command recursively searches the given db path with the -ucdbname file name pattern and generates the list of files to merge.
- **-genlistfrom <pathdir_or_list>...**
(optional) Generates the list of files to merge by recursively searching the specified database path(s), at the location specified by the -filelist option, for the file pattern specified by -dbname.
You can specify multiple database paths as a space-separated list. If a list of databases is entered, the list must be enclosed in 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 processes on the selected grid type. You can use the -j option in conjunction with -gridslots. The -j option determines how many files each parallel process will merge, and -gridslots determines how many parallel processes can run on the grid at a time.
- **-hostlist <pattern>**
(optional) Specifies the host list, or the file containing the host list (Default: ""). The number of hosts provided in the list determines the maximum number of parallel jobs to run. Specify the file containing list of hosts as "host1[:<n1>][, host2[:]<n2>] ...]". The file can contain multiple lines with this syntax. The vcover command reads this file and expands the hostlist as a space-separated list; "host1:2, host2:3" becomes "host1 host1 host2 host2 host2", and the number of maximum jobs will be 5.
- **-j <val>**
(optional) Specifies the maximum number of parallel merge jobs (Default: 10) to run. The maximum number of parallel jobs is used when splitting the filelist to determine the number of files merged by each parallel process. For 'rsh' -runmode type, this value is calculated from -hostlist.
- **-log <filename>**
Specifies where to save a log file for the command.
- **-maxrerun <integer>**
Specifies the maximum number of times to automatically rerun job queue, timeouts or merge failure processes.

 <integer> — Default: 10.
 0 — Disables reruns for all failure types

- **-mergererun**
Specifies that merge failure processes are allowed to rerun automatically.
- **-mergetype <totals/testassociated/preservetestcounts>**
(optional) Specifies 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 to use for time-outs, measured between the time a merge process starts and the time the process finishes. The value (<val>) is the specific maximum runtime value from all of the first or second level parallel merge processes. The runtime for each parallel merge process depends on the size of single database files, the number of total files to merge, and the number of parallel processes used. (Default: 10000).
- **-outname <name>**
(optional) Specifies the merged output file name (Default: "merge.ucdb" for UCDB and "merge" for Coverstore).
- **-qtimeout <val>**
(optional) Specifies the queue timeout value. This value is used for time-outs and is measured from the time that a merge process is launched to the time it actual starts to execute. (Default: 5000).
- **-quiet**
Returns minimal messaging, which includes start and end output.
- **-rmdb <path>**
(optional) Specifies a user-defined RMDB file. The vcover parallelmerge command auto-generates the RMDB file in \$cwd/TEMP_MERGE_DIR based on the provided vcover options. If you want to use a customized RMDB file, you can copy the auto-generated RMDB file, modify it as needed, and include it with this option.
- **-runmode <local/rsh/lsh/sge/uge/rtda>**
(optional) Specifies the type of parallel merge run mode. The default is local.
- **-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).
Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the default settings "cmd,msg,time".

<feature>

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.

Note

 Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

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.

- **-tempdir <dir_path>**

Specifies a temporary directory for intermediate output. Default: \$cwd/TEMP_MERGE_DIR

- **-teststatus <all/passed/failed>**

(optional) Merges only tests that have the status specified in the TESTSTATUS value in UCDB file (Default: all). When you specify passed or failed, vcover scans the provided or generated filelist and merges only those files with the given teststatus.

- **-verbose**

Returns full messaging, including a full log of all parallel processes.

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 databases in the directory dir1 (UCDB mode):

```
vcover parallelmerge -genlistfrom dir1
```

Related Topics

[vcover merge](#)

[Parallel Merge \[Questa SIM User's Manual\]](#)

vcover ranktest

Ranks coverage data contained in specified tests, or test groups, according to their contribution to cumulative coverage.

Syntax

```
vcover ranktest [<ranktest_options>] {<UCDB_inputfile1> [...] <UCDB_inputfileN>} | -inputs <file_list>

<ranktest_options> =
  [-32 | -64] [-algorithm {greedy | quick}]
  [-nocompulsoryordering | -compulsorysorting | -compulsoryranking]
  [-error <msg_number>[,<msg_number>,...]]
  [-fewest | -cputime | -simtime [-goal [<coverage_type>] <percentage>]]
  [-groupby <attribute> [[-groupfilter <regex>] [-norun] [-r[ecursive]]]]
  [-iterative | -testassociated]
  [-keepmergefile <filepath>]
  [-logfile <filename> | -l <filename>]
  [-maxcpu <real_num_in_seconds>] [-maxtests <int>] [-metric {aggregate | total}]
  [-modelsimini <path/modelsim.ini>]
  [-note <msg_number>[,<msg_number>,...]]
  [-[no]parallelism] [-path <path> | -du <du_name> | -plansection <path>]
  [-precision <int_num>] [-[w]prof=<filename>] [-proftick=<integer>]
  [-quiet]
  [-rankfile <filename>]
  [-stats [=+ | -]<feature>[,[+ | -]<mode>]] [-suppress <msg_number>[,<msg_number>,...]]
  [-warning <msg_number>[,<msg_number>,...]] [-weight <coverage_type> <integer>]

<linux_platform_only_options> =
  [-j <number_of_parallel_processes>] [-[no]parallelism]

<coverage_type> =
  [[-assertfailure] [-assertion] [-code {b | c | e | f | s | t}...]] [-codeAll] [-cvg] [-directive]]
```

Description

The tests or groups to rank are taken from the UCDB files you specify as inputs. These can be merged or unmerged UCDBs as well as merged coverstores. If merged, they must have been created with the vcover merge -testassociated argument.

The command writes output to stdout (screen) in two lists of tests; contributing tests in *ranktest.contrib*, and non-contributing tests in *ranktest.noncontrib*. The output also lists any missing tests. The tests are written to the screen or to a file you specify (with -log), in the order shown in [Table 3-9](#)

Table 3-9. Order and Type of Ranked Tests

Contributing, compulsory	Mandatory tests, tests which need to be run regardless of achieved coverage.	S o r t e d b y t o t a l c o v e r a g e %
-----------------------------	--	--

Table 3-9. Order and Type of Ranked Tests (cont.)

Contributing, noncompulsory	Tests providing coverage not provided by any previous test.	S o r t e d b y t o t a l c o v e r a g e %
Non-contributing	Redundant tests, providing no incremental coverage.	N o t s o r t e d

To rank a specific coverage item, design unit, or testplan section within the hierarchy of your design, use -path, -du or -plansection.

Use the -groupby argument to specify an attribute within a UCDB to use to group tests. You can also specify a regular expression with the -groupfilter <regex> argument, to use to match on a specific part of an attribute value.

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 the algorithm to use to produce the ranked list of tests. The default algorithm is greedy, which produces the shortest list of tests contributing to coverage, arranged in order of highest incremental contribution to least. The quick algorithm provides a quick ordering of tests, which may not result in the shortest list. The quick list uses the incremental order, highest to lowest, from the first parse coverage calculations, merging the highest increasing coverage tests first until the maximum coverage is reached.

- **-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.

- **-assertion**

Specifies ranktest for assertion data. This argument is not compatible with -metric total: if both are explicitly set, ranking halts 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 can be specified with each -code argument (example: “-code bcest”). This argument is not compatible with -metric total: if both are explicitly set, ranking halts 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 halts and an error is issued.

- **-nocompulsoryordering | -compulsorysorting | -compulsoryranking**

Specifies how the command ranks compulsory tests, when present.

-nocompulsoryordering — (default) Lists 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 displays 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. The GUI calculates and displays the accumulated coverage data and incremental coverage data for compulsory tests. This option has performance second to the -nocompulsoryordering option.

-compulsoryranking — Ranks compulsory tests fully with respect to each other. Displays compulsory tests in the detailed ranktest listing according to this order. The GUI calculates and displays the accumulated coverage data and incremental coverage data for compulsory tests. This option can have the worst performance of the three options.

- **-cputime**

(optional) Specifies to select tests for ranking by 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 halts 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 halts 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 you do not specify a name, defaults to work. 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. Specify multiple message numbers as a comma-separated list.

Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to [error](#) and “[Message Severity Level](#)” in the User’s Manual for more information.

- **-fewest**
(optional) Specifies to select tests for ranking by the largest coverage gain with the fewest number of tests. Mutually exclusive to the **-cputime** and **-simtime** arguments. Default.
- **-groupby <attribute>**
(optional) Specifies the attribute within a UCDB test record to use to group tests by.
- **-groupfilter <regex>**
(optional) Specifies a regular expression match to apply to the attribute being grouped. Must be used with the **-groupby** argument.
- **-inputs <file_list>**
(optional) Specifies a file containing ranktest arguments. The file can contain a list of UCDB files to rank.
 <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 to use during the ranktest operation.
- **-goal [<coverage_type>] <percentage>**
(optional) Specifies to rank the tests to achieve the specified coverage goal. This argument is not compatible with **-metric total**: if both are explicitly set, ranking halts and an error is issued.
 <coverage_type> — specifies to apply the goal to only the specified type of coverage; otherwise, the goal is applied to all types of coverage.
 Valid values for <coverage_type> are: **-assertion**, **-assertfailure**, **-code {b | c | e | f | s | t}...**, **-codeAll**, **-cvg** or **-directive**.
 <percentage> — an integer value; the default is 100.

You can specify the **-goal** argument multiple times, as shown in this example:

```
vcover ranktest -goal -assertion 90 -goal -code bcest 95
```

- **-keepmergefile <filepath>**
(optional) Specifies to preserve the merge file corresponding to the ranking. 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 is a forward slash (/).

- **-logfile <filename> | -l <filename>**
(optional) Specifies a file in which to save the ranked results. The output defaults to stdout, and is echoed to any explicitly supplied log file. The output includes the 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 allow. The ranking process stops if the specified number of seconds is exceeded.

 <real_num_in_seconds> — Any integer greater than -1.0 where the default is -1.0 (no limit).
- **-maxtests <int>**
(optional) Specifies the threshold for the maximum number of tests to rank. When the threshold is exceeded, the ranking operation stops.

 <int> — Any positive integer.
- **-metric {aggregate | total}**
(optional) Specifies the metric to use for ranking.

total —

(Default, unless any of the following arguments are used: -goal, -weight, -assertion, -assertfailures, -cvg, -directive, -codeAll, or -code)

Produces values consistent with the UCDB totals obtained with coverage analyze, which appear in Total Coverage numbers. Refer to "["Calculation of Total Coverage"](#)" in the User's Manual for details.

aggregate — Produces aggregate values based on each individual coverage type: values are not likely to be consistent with totals produced by any other means.

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 you use the -metric aggregate argument, 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"](#)" in the User's Manual 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 name. On Windows systems the path separator is a forward slash (/).

- **-norun**

Display groups without ranking. Allows you to see grouping before running the ranking process. Must be used with the **-groupby** argument.

- **-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. Specify multiple message numbers as a comma-separated list.

Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to [note](#) and [“Message Severity Level”](#) in the User’s Manual for more information

- **-[no]parallelism**

(optional) Allows or disables parallel ranking of multiple processes. Not implemented for Windows platforms.

- **-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 is 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 is 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>. Customer Support uses output from these arguments 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.
- **-r[ecursive]**
Enables recursive ranking of items within a group selected with -groupby and -groupfilter so they will, in turn, be ranked against one another. Must be used with the -groupby argument.
- **-rankfile <filename>**
(optional) Specifies the name of the ranktest file being created. If not specified, defaults to *ranktest.rank*. Can be specified with the [vcov_r 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 to select tests for ranking by 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).
Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the default settings "cmd,msg,time".
<feature>
 - 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.

Note



Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

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 to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

<msg_number> — A number identifying a specific message. Specify multiple message numbers as a comma-separated list.

Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to “[suppress](#)” in the User’s Manual for more information.

- **-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.

- **<UCDB_inputfile1> [... <UCDB_inputfileN>]**

(required unless the -inputs file argument is used) Specifies the name of two or more non-merged UCDB file(s) to rank. Can be in the form of either a UCDB or coverstore[:test] file.

- **-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. Specify multiple message numbers as a comma-separated list.

Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to [warning](#) and “[Message Severity Level](#)” in the User’s Manual 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 halts and an error is issued.

<coverage_type> — specifies to apply the goal to only the specified type of coverage; otherwise, the goal is applied to all types of coverage. Valid values for <coverage_type> are:

[-assertion](#), [-assertfailure](#), [-code {b | c | e | f | s | t}...](#), [-codeAll](#), [-cvg](#) or [-directive](#).

<integer> — Any non-negative integer.

You can specify the **-weight** argument multiple times, as shown in this example:

```
vcover ranktest -weight -assertion 10 -weight -code bcest 5
```

Examples

- Rank the first ten tests in mergefile.ucdb, in a ‘testassociated manner,’ and output to a rankfile named my_file.rank with decimal point precision of 4.

```
vcover ranktest -testassociated -maxtests 10 -rankfile my_file.rank -precision 4  
mergefile.ucdb
```

- Merge a set of UVM tests with different random seeds, then group these into 2 groups so you can see which test (regardless of seed value) is providing the most value. Assume the following attributes in 10 different UCDB files (where *TESTNAME* is <testname>_<seed>):

TESTNAME = testA_1	TESTNAME = testB_1
TESTNAME = testA_2	TESTNAME = testB_2
TESTNAME = testA_3	TESTNAME = testB_3
TESTNAME = testA_4	TESTNAME = testB_4
TESTNAME = testA_5	TESTNAME = testB_5

You can merge the tests, then group and rank them, with the following commands:

```
vcover merge -out merge.ucdb <inputs>
```

```
vcover ranktest merge.ucdb -groupby TESTNAME -groupfilter (<.*>)*
```

The parentheses in the filter expression are used to define the group (and subsequently the group name). The result is a ranked output containing testA and testB.

Related Topics

[vcover merge](#)

[vcover stats](#)

[Code Coverage \[Questa SIM User's Manual\]](#)

vcover remove

Enables you to create a copy of a previously merged testplan UCDB, except that the new copy does not include testplan data, links, or tags.

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 new UCDB, to be created without the testplan section. When you use -out, you can specify the <outfile> after the <infile>.
- **<infile>**
(required) The name of the original UCDB file with the testplan section to remove.
- **-[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>. Customer Service uses output from these arguments 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).

Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the default settings "cmd,msg,time".

<feature>

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.

Note

 Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

Modes can be set for a specific feature, or set globally for all features. To add or subtract a mode for a specific feature, use 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 kilobyte 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 remove -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

Generates textual or HTML output of coverage statistics or exclusion, extracting the information from a previously saved code coverage or functional coverage run, and writing it to a specified file.

Syntax

```
vcover report [<coverage_arguments>] {<ucdb or coverstore_tests> = <ucdb_file> |  
    <coverstore>[:test]}
```

```
vcover report [-version] <input_ucdb>
```

where <coverage_arguments> can be categorized into the following groups:

Coverage Arguments

```
vcover report <rpt_args> <ucdb_or_coverstore_test>
```

<rpt_args> =

```
[ -32 | -64 ] [ -above <percent> ] [ -append ] [ -below <percent> ] [ -binrhs ] [ -bydu ] [ -byfile ] [ -  
byinstance ] [ -details [-dumptables] ] [ -file <filename> ] [ -flat [-primarykey <type | path |  
value> [-secondarykey <type | path | value>]] ] [ -memory ] [ -modelsimini <path/  
modelsim.ini> ] [ -nocomment ] [ -nomissing ] [ -pa ] [ -pacombined ] [ -totals ] [ -nozeroweights ]  
[ -precision <int> ] [ -[w]prof=<filename> ] [ -proftick=<integer> ] [ -recursive [-depth <n>] ] [ -  
showambiguity ] [ -stats [=+ | -]<feature>[,<mode>] ] [ -testextract  
<test_name_or_pattern> ] [ -xml ] [ -suppress <msg_number>[,<msg_number>, ...] ] [ -error  
<msg_number>[,<msg_number>, ...] ] [ -note <msg_number>[,<msg_number>, ...] ] [ -  
warning <msg_number>[,<msg_number>, ...] ] [ -zeros ]
```

Arguments for creating HTML output from a Coverage Database

```
vcover report -html <input_ucdb> [ -code [bcesf[t|x]] ] [ -assert ] [ -binrhs ] [ -cvg ] [ -directive ]  
[ -details[=abcdefgst] ] [ -du <du_name> ] [ -hidecvginsts | -hidecvginstspi0 ] [ -instance  
<path> ] [ -notimestamp ] [ -nozeroweights ] [ -plansection <path> ] [ -showcvggoalpcnt ] [ -  
showexcluded ] [ -source ] [ -summary ] [ -testhitdata[=<int>] ] [ -testhitdataAll ] [ -testdetails ]  
[ -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 | x}... ] [ -codeAll ] [ -concurrent | -immediate ] [ -  
covered ] [ -cvg ] [ -directive ] [ -failed ] [ -filter <pattern> ] [ -lang sva | psl | vhdl ] [ -severity info  
| note | warning | error | failure | fatal ] [ -testattr ] [ -unattempteddimmed ]
```

Note

 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

```
vcover report [-coverenhanced] [-noannotate] [-library <libname>] [-du <du_name>] [-package <pkgnname>] [-source <filename>] [-instance <path>]
```

Note

 Code coverage arguments have no effect on assertion or functional coverage.

Ranked Coverage Arguments

```
vcover report [<rankfile>] [-attribute=<attr_name>[+<attr_name>]] [-attributeAll] [-showbincounts] [-showincr] [-showucdb] [-assert] [-code {b | c | e | f | s | t | x}... ] [-codeAll] [-cvg] [-directive] [-error <msg_number>[,<msg_number>,...]] [-nononcontrib] [-nosummary] [-note <msg_number>[,<msg_number>,...]] [-stats [=+|-]<feature>[,[+|-]<mode>]] [-suppress <msg_number>[,<msg_number>,...]] [-warning <msg_number>[,<msg_number>,...]]
```

Arguments for Creating HTML Reports for Ranked Results

```
vcover report -html [-attribute=<attr_name>[+<attr_name>]] [-attributeAll] [-details[=abcdefgpst]] [-htmldir] [-nobins] [-nographs] [-nononcontrib] [-nosummary] [-precision] [-stats [=+|-]<feature>[,[+|-]<mode>]]
```

Coverage Exclusion Arguments

```
vcover report -excluded [-pragma] [-user] [-nocomment] [-noexcludedhits] [-code {b | c | e | f | s | t | x}... ] [-instance <path>] [-file <filename>] [-append]] [-adaptive]
```

Toggle-specific Coverage Arguments

```
vcover report [-toggles] [-duplicates] [-verbose] [-all] [-portmode {input | output | inout | internal}]
```

For a Report of Non-collected Coverage Items

```
vcover report -nocollect [-file <filename> [-append]]
```

Covergroup-specific Arguments

```
vcover report [-binrhs] [-cvg] [-hidecvginsts | -hidecvginstspi0] [-nocrossbinsummary] [-nocvbinsummary] [-ignorebin] [-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...}] [-descendants] [-directive] [-fixedtotalcov] [-verbose] [-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 —

```
vcovre 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:

```
vcovre report -trend <trend_ucdb> {-bydu | -byinstance | -byplan} [-du <duname>] [[-instance |  
-path] <path>] [-plansection <path>] [-precision <int>] [-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, or Text format of a trending database; used for the purpose of analyzing coverage over time.

Description

This command allows you to produce reports "offline" (without having to load a simulation) and provides arguments to refine the report output. It also enables you to report ranked results, and presents the data in the GUI, and in text or HTML.

Tip

 HTML reports are created with the -html option, which is compatible only with the listed -html arguments. 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:

- vcov^re report <path_to_cover_directive/covergroup>

The report orders information by file, and also displays code coverage data from generate blocks.

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, you must also include the -details argument. To report extended

toggle coverage, you must compile (vlog/vcom) with the -code x argument, then use vcover report with -code t.

Arguments

- <input_ucdb>
(required) Specifies the previously saved code coverage file on which 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) Specifies to include only objects with coverage values above the specified percentage in the output. Includes coverage of coverpoints and crosses, not covergroups. Refer also to -below.

 <percent> — An integer between 0 and 100.
- -adaptive
When used with the -excluded argument, generates an adaptive exclusion DO file.
- -all
(optional) When reporting toggles, creates a report that lists both toggled and untoggled signals. Reports counts of all enumeration values. Not a valid option when reporting on a functional coverage database.
- -append
(optional) Appends the report data to the named output file.
- -assert
(optional) Adds a column for assertion data to the coverage report.
- -attribute=<attr_name>[+<attr_name>]
Enables the display of any additional attribute (user-defined or tool-defined) stored in the UCDB. Use the plus sign (+) to separate multiple attribute names.
- -attributeAll
Enables display of all additional attributes stored in the UCDB.
- -below <percent>
(optional) Specifies to include only objects with coverage values below the specified percentage 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, for covergroups, to include a column 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 iterates through all design units/modules in the design and reports coverage data for each. Each design unit report is the merged result of all instances of that module and is sorted by design unit name. You can also report coverage data for a specific design unit with the [-du <du_name>](#) argument.
- **-byfile**
(optional) Writes out a coverage summary for each source file in the design. This is the default report setting.
- **-byinstance**
(optional) Writes out a coverage summary for all instances and packages. Can be used with the [-recursive \[-depth <n>\]](#) argument to report on all instances recursively. The default setting is [-byfile](#).
- **-code {b | c | e | f | s | t | x}...**
(optional) Specifies which code coverage statistics to include in the report, and adds a column for each specified coverage type. If no arguments are passed to [-code](#), the report includes statistics for all categories that were 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.
- x — Include extended 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](#).

- **-codeAll**
(optional) Equivalent to [-code bcestf](#). Adds a column to the report for all code coverage types found in the UCDB. Not compatible with [-metric total](#): if both are explicitly set, ranking halts and an error is issued.

- **-concurrent | -immediate**
(optional) Selects only the indicated assertions. These are mutually exclusive options. If you specify neither switch, 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. The ancestors of any covergroup item that survives the applied covergroup filters are displayed in the report and the GUI.
- **-coverenhanced**
(optional) Enables non-critical functionality, which can change the appearance or content of coverage metrics. This argument has an effect only in letter releases (10.0a, 10.0b, and so on). It is not necessary with major releases (10.0, 10.1, and so on), which enable all coverage enhancements present in previous letter release streams by default. Bug fixes important to the correctness of coverage numbers are always enabled by default, and do not need -coverenhanced. For details on enhancements enabled in the current release, refer to the product release notes, and search on the string "coverenhanced".
- **-cvg**
(optional) Adds a column for covergroup data to the coverage report.
- **-details[=abcdefgpst]**
Enables display of individual coverage metrics. You can selectively include only **assertions**, **branches**, **conditions**, cover **directives**, **expressions**, **FSMs**, **covergroups**, **testplan**, **statements**, and **toggles**, or any combination thereof.

Note

 The function of the -details argument changes with the context in which it is used.
Here it is used for generic coverage reporting.

- **-details [-dumptables]**
(optional) Includes details associated with each coverage item in the output for FEC coverage. By default, details are not provided.
-dumptables — (optional) forces printing of condition and expression truth tables, even though fully covered.

Note

 The function of the -details argument changes with the context with which it is used.
Here it is used for creating HTML output from a UCDB.

- **-directive**
(optional) Adds a column for cover directive data to the coverage report.

- **-du <du_name>**

(optional) Reports coverage statistics for the specified design unit. Any parameterized instances are considered to match the specified design unit. This coverage is the combined "local" coverage of a flat list of all instances of the 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, defaults to work. 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. Specify multiple message numbers as a comma-separated list.

Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to [error](#) and [“Message Severity Level”](#) in the User’s Manual 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 appear, by default, in the exclusion report created with -excluded. You can also use the GUI to add, modify, and remove comments.

- **-failed**

(optional) Reports only failed assertions/cover directives.

- **-file <filename>**

(optional) Specifies an output file name for the report. The default is to write the report to the Transcript window.

<filename> — A user specified string. Environment variables can 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 match in the search. Allows wildcards.
- **-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, assertions, coverdirectives, covergroups, 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, assertions, coverdirectives, covergroups, 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.
- **-htmldir**
Specifies the output HTML directory for the report.
- **-html <input_ucdb> [-code [bcesf[t|x]] [-assert] [-binrhs] [-cvg] [-directive] [-details[=abcdefgst]] [-du <du_name>] [-hidecvginsts | -hidecvginstspi0] [-instance <path>] [-notimestamp] [-nozeroweights] [-plansection <path>] [-showcvggoalpcnt] [-showexcluded] [-source] [-summary] [-testhitdata[=<int>]] [-testhitdataAll] [-testdetails] [-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 generated file.
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 <du_name> — (optional) Includes information for only the specified design unit in the HTML report; the default is to include all design unit information. You can specify more than one instance by using the -du option multiple times in the command, or by using wildcards (*).
- hidecvginsts | -hidecvginstspi0 — (optional) Removes all covergroup instances from the HTML report, or only those having per_instance set to 0.
- instance <path> — (optional) Includes only the specified instance in the HTML report; the default is to include all instances. The instance(s) appears in the report recursively, including any downward hierarchy. You can specify more than one instance by using the -instance option multiple times in the command, or by using wildcards (*).
- notimestamp — (optional) Prevents timestamp information from appearing in the report.
- nozeroweights — (optional) Prevents items with zero weights from appearing in the report, including all zero-weighted coverage items and testplan sections. Has no effect on coverage numbers.
- plansection <path> — specifies particular verification plan sections to which the report is applied.
- precision <int> — (optional) Sets the precision for the values displayed in HTML report.
- showcvgoalpcnt — (optional) Displays a column for Goal % in the covergroups table. If you do not specify this argument, the value of Goal % is displayed in the tooltip for each covergroup.
- showexcluded — (optional) Shows the excluded items in the HTML report; the default is to not show excluded items.
- source — Generates the annotated source. The default is to suppress source files for HTML. This argument is required in order to access source code related data inside the generated HTML report.
- summary — (optional) Includes only the top summary page, the testplan summary page, and the list of tests run in the generated report.
- testhitdata[=<int>] — (optional) Enables generation of tables containing test-bins-hit information for <int> number of tests that hit a bin. The default when you don't specify an integer <int> is 10. The -testhitdata argument produces tables for a merged UCDB (unless merged explicitly with vcover merge -totals) in the HTML report. The tables contain information regarding which bins were hit by which tests, and include bin numbers, hyperlinked to the relevant test data.
- testhitdataAll — (optional) Enables generation of a table containing test-bins-hit information for all tests that hit a bin.

- testdetails — (optional) In addition to the test summary page, this argument generates a page of information for each test that was run. The default is to exclude these pages.
- 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 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). Allows multiple -instance <path> specifications. This coverage is the combined "local" coverage of the instance(s).

 <path> — Specifies either an absolute or relative path to the instance(s). On Windows systems the path separator is a forward slash (/).
- -lang sva | psl | vhdl
(optional) Specifies assertions of a specific language (SVA, PSL, or VHDL). You can use this option multiple times to specify multiple languages. If you do not specify -lang, 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, including information on 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 to replace 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 name. On Windows systems the path separator is a forward slash (/).
- **-noannotate**
(optional) Removes source code from the output report. Valid for code coverage only. Not applicable with -xml argument.
- **-nobins**
Disables reporting of bins for individual coverage types. Coverage is reported as “percent covered” instead of the default “number of bins hit/total bins.”
- **-nocomment**
(optional) 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 -out argument, and optionally, -append. Used with -out 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.
- **-nocvbinsummary**
(optional) Removes cross bins for each coverpoint and cross from the report output.
- **-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).

- **-nographs**
Prevents generation of the pie and line chart at the top of the report in the summary section.
- **-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.
- **-nononcontrib**
Prevents generation of list of non-contributing tests.
- **-nosummary**
Prevents generation of initial summary section (the graphs). (The total coverage table is always generated.)
- **-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. Specify multiple message numbers as a comma-separated list.

Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to [note](#) and “[Message Severity Level](#)” in the *Questa SIM User’s Manual* 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.
- **-pa**
Generates the Power Aware coverage report, which contains information about Power Aware coverage. Use this argument with the -details, -verbose, -assert, -file, -html, and -xml arguments. Refer to “[Generating the Power Aware Coverage Report](#)” in the “*Power Aware Simulation User’s Manual*” for more information.
- **-pacombined**
Generates the basic and advanced combined coverage report, which contains information about Power Aware and non-Power Aware coverage. Use this argument with the -details, -verbose, -assert, -file, -html, and -xml arguments. Refer to “[Generating the Basic](#)

“[Combined Coverage Report](#)” and “[Generating the Advanced Combined Coverage Report](#)” in the *Power Aware Simulation User’s Manual* for more information

- **-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**
Specifies how many numbers after the decimal point to display.
- **-precision <int>**
(optional) Reports on the instance specified with -instance, and every included instance, recursively. Can also be used with -details and -totals.
- **-[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>. Customer Support uses output from these arguments for debugging purposes.
- **-proftick=<integer>**
(optional) Sets the time interval between the profile data collections. Default = 10.
- **<rankfile>**
Specifies the name of a ranktest file created with the vcover ranktest command.
- **-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. (Refer to [SVCovergroupSampleInfo](#) in the User’s Manual.) This argument filters to include covergroups whose sample count matches the criteria given in <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 spaces allowed) sample count is in the range lower-upper. There is only 1 -sample option allowed per invocation.
- -severity info | note | warning | error | failure | fatal
 - (optional) Specifies the assertion severity level. When you specify -severity, 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).
 - -showbincounts
 - Enables the visibility of bin counts for many coverage items (for example, # of bins hit, total # of bins). Works only when <rankfile> is given.
 - -showincr
 - (optional) Enables visibility of the ‘*Incr’ columns for each of the enabled coverage types. Works only when <rankfile> is given.
 - -showucdb
 - (optional) Enables output of the ucdb filename for each ranked item. You must list all other arguments before -showucdb in the command line. Works only when <rankfile> is given.
 - -source <filename>
 - (optional) Writes a summary of statement coverage data for a specific source file.
 - <filename> — Specifies path and name of the file. On Windows systems the path separator is a forward slash (/). Environment variables can 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).

Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this

switch adds or subtracts features and modes from the default settings "cmd,msg,time".

<feature>

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.

Note

 Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

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 to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

<msg_number> — A number identifying a specific message. Specify multiple message numbers as a comma-separated list.

Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to [suppress](#) in the User's Manual for more information.

- -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. You can apply multiple -testextract arguments in same command. This argument is compatible with reports generated in plain text and XML formats only; HTML reports are not supported. With 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 cannot 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. The default, without this argument, is for the report to write out an instance summary for each of the instances. The report prints only one summary if you use the -totals option. 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 the toggles 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](#).

- **-trend <trend_ucdb>**

```
{-bydu | -byinstance | -byplan} [-du <duname>] [[-instance | -path] <path>]
[-plansection <path>] [-precision <int>] [-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 can 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, shows only the local instance coverage numbers.
 - 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, but 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 <%> — graphs only 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 exports 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, shows only the local instance coverage numbers.

-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...}] [-descendants] [-directive] [-fixedtotalcov]
[-verbose] [-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.

-fixedtotalcov — displays the actual total coverage values in a Coverage Summary graph, irrespective of selected coverage types in that graph. If no coverage type is selected, then only the total coverage is displayed.

([-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.

-precision <int> — sets precision for output.

-showattrswithcoverage <attr1{,<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 data, to be displayed on the same graph as coverage. For example: comparing the number of passed, failed, and so on, 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, shows only the local instance coverage numbers. Valid only 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 is 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 to use the value of SVCrossNumPrintMissingDefault in the report. (Refer to [SVCrossNumPrintMissingDefault](#) in the User's Manual.) 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. The list includes 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 the 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. You can specify multiple message numbers as a comma-separated list.
Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to [warning](#) and [“Message Severity Level”](#) for more information.
- **-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](#) in the User’s Manual 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 -instance options
For covergroups, this argument applies to coverpoint and cross bins only.
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 -out 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 follow statement statistics, and will be presented in four columns: line, column, true branch count, false branch count.

vcover report -details -out save.cov input.ucdb

- Write code coverage details of one specific instance to *save.cov*.

vcover report -details -instance /top/p -out 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 -out myreport.txt input.ucdb

- Write a list of statements with zero coverage to *myzerocov.txt*.

vcover report -zeros -out 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 \[Questa SIM User's Manual\]](#)

[Analyzing Trends \[Questa Verification Management User's Manual\]](#)

vcover stats

Prints summary statistics for previously saved code coverage databases to *stdout*.

Syntax

```
vcover stats [-32 | -64] [-assert] [-cvg] [-directive] [-code {b | c | e | f | s | t}...]
[-codeAll] [-nofec] [-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] ...}
```

Description

You can invoke vcover stats from the Questa SIM GUI or the command line.

The vcover stats command creates coverage statistics output that is equivalent to the output from invoking vcover report -totals -byinstance.

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 details.
- **-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 options are:

- 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 the 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. Specify multiple message numbers as a comma-separated list.
Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to [error](#) and "[Message Severity Level](#)" in the User's Manual 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. You can specify multiple message numbers as a comma-separated list.
Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to [error](#) and "[Message Severity Level](#)" in the User's Manual 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 you use coverstore, you can optionally specify a comma-separated list of tests to include in the stats. You can include multiple pathnames and use wildcards.
- **-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 to replace 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 name. On Windows systems the path separator is 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:”) 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. Specify multiple message numbers as a comma-separated list.
 Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to [note](#) and “[Message Severity Level](#)” in the User’s Manual for more information
- **-precision <int>**
 (optional) Sets the decimal precision for printing coverage information. Valid values for `<int>` are from 0 to 6 and the 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>`. Customer Service uses output from these arguments 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).
 Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the default settings "cmd,msg,time".

<feature>

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.

Note

 Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

Modes can be set for a specific feature, or set 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 kilobyte 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 to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

<msg_number> — A number identifying a specific message. Specify multiple message numbers as a comma-separated list.

Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to [suppress](#) in the User's Manual for more information.

-
- **-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. Specify multiple message numbers as a comma-separated list.

Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to [warning](#) and “[Message Severity Level](#)” in the User’s Manual for more information.

Examples

- Using the -stats argument, 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 <ucdb_file>

- 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 <ucdb_file>

Related Topics

[vcover merge](#)

[Code Coverage \[Questa SIM User's Manual\]](#)

[vcover ranktest](#)

vcover testnames

Displays the test names in the specified UCDB file or merged UCDB file.

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>,...]]]
```

Description

This command is useful for the -testextract argument of a [coverage analyze](#) or [vcover report](#) command, because -testextract requires the test name.

(By default, a test's name is the same as the UCDB file name. You can change the test name with the -testname argument to the [coverage save](#) command.)

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. Specify multiple message numbers as a comma-separated list.

Edit the error variable in the *modelsim.ini* file to set a permanent default. Refer to [error](#) and “[Message Severity Level](#)” in the User’s Manual 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 to replace 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 name. On Windows systems the path separator is 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. Specify multiple message numbers as a comma-separated list.

Edit the note variable in the *modelsim.ini* file to set a permanent default. Refer to [note](#) and “[Message Severity Level](#)” in the User’s Manual 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>. Customer Support uses output from these arguments 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).

Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the default settings "cmd,msg,time".

<feature>

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.

Note

 Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

Modes can be set for a specific feature, or set 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 kilobyte 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 to suppress. You cannot suppress Fatal or Internal messages (those without numbers).

<msg_number> — A number identifying a specific message. Specify multiple message numbers as a comma-separated list.

Edit the suppress variable in the *modelsim.ini* file to set a permanent default. Refer to [suppress](#) in the User's Manual for more information.

- **-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. Specify multiple message numbers as a comma-separated list.

Edit the warning variable in the *modelsim.ini* file to set a permanent default. Refer to [warning](#) and ["Message Severity Level"](#) in the User's Manual 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 \[Questa SIM User's Manual\]](#)

[Verification Management Browser \[Questa SIM GUI Reference Manual\]](#)

[vcover merge](#)

[vcover ranktest](#)

[**vcover stats**](#)

vdel

Deletes a design unit from a specified library.

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 deleted libraries. You are not prompted for confirmation.

- **-allsystemc**
(optional) Deletes all SystemC modules in a design from the working directory.
- **-dpiobj [<object_info>]**
(optional) Deletes auto-compiled DPI object files.

<object_info> — Specifies the type of object files to remove, as reported in the output of the vdir -obj command. The object file types are:
 - <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 to remove all object files reported in the output of the vdir -obj command.
- **-lib <library_path>**
(optional) Specifies the location of the library containing the design unit to delete. The default is to delete the design unit from the work library.

<library_path> — The logical name or pathname of the library.
- **-modelsimini <path/modelsim.ini>**
(optional) Loads an alternate initialization file to replace 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 name. On Windows systems, the path separator is 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 in the output of the vdir -obj command. The directory types are:

<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 to remove all directories reported in the output of the vdir -obj command.

- <primary> [<arch_name>]

(required unless -all is used) Specifies the entity, package, configuration, or module to delete.

This option is not supported for SystemC modules.

<arch_name> — Specifies the name of an architecture to delete. If omitted, all 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

Lists the contents of a design library and checks the compatibility of a vendor library.

Syntax

```
vdir [-l | [-prop <prop>]] [-r] [-obj] [-all | [-lib <library_name>]] [<design_unit>] [-modelsimini  
<path/modelsim.ini>]
```

Description

This command lists SystemC modules that are exported with the SC_MODULE_EXPORT() macro.

If vdir cannot read a vendor-supplied library, the library may not be compatible with Questa SIM.

Arguments

- **-all**
(optional) Lists the contents of all libraries included in the Library section of the active *modelsim.ini* file. Refer to [modelsim.ini Variables](#) in the User's Manual for more information.
- **<design_unit>**
(optional) Specifies the design unit to search for in the specified library. If the design unit is a VHDL entity, its architectures are listed. The default is to list all entities, configurations, modules, packages, and optimized design units in the specified library.
- **-l**
(optional) Prints the version of vcom/vlog/sccom used to compile each design unit, and any compilation options. Also prints the object-code version number that indicates which versions of the executable and Questa SIM are compatible.
- **-lib <library_name>**
(optional) Specifies the logical name or the pathname of a library to list. The default is to list the contents of the work library.

 <library_name> — The logical name or pathname of a library.
- **-modelsimini <path/modelsim.ini>**
(optional) Loads an alternate initialization file to replace 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 name. On Windows systems, the path separator is 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 3-10](#). If you do not specify a value for <prop>, an error message displays.

Table 3-10. Design Unit Properties

Value of <prop>	Description
archcfg	configuration for arch
body	needs a body
cmptime	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

Encrypts Verilog and SystemVerilog code.

Syntax

```
vencrypt <filename> [<filename> . . .] [-allow_veloce] [-common <filename>] [-  
data_method=[aes128|aes192|aes256]] [-d <dirname>] [-e <extension>][-f <cmdfile>] [-hea  
<headerfile>] [-keyring <directory>] [-logfile <logfile> | -l <logfile>] [-o <outputfile>] [-p  
<prefix>] [-quiet] [[-stats [=+ | -]<feature>,[+ | -]<mode>]] [toolblock  
<keyfile>[,<rightsfile>]] [-wholefile]
```

Note

 The following arguments are only for the automatic encryption of Verilog. SystemVerilog is not supported.

```
vencrypt [-autotprotect] [-autoprotect_all] [-auto2protect] [-auto3protect] [-auto5protect] [-  
pli_unprotected]
```

Description

The vencrypt command does not pre-process code before encryption, so macros and other `directives are unchanged, allowing you to deliver encrypted IP with undefined macros and `directives.

Upon execution of this command, the filename extension changes to *.vp* for Verilog files (*.v* files) and *.syp* for SystemVerilog files (*.sv* files).

If the vencrypt utility processes the file (or files) and does not find any encryption directives, it reprocesses the entire file using the following default encryption:

```
`pragma protect version = 1  
`pragma protect key_keyowner = "Mentor Graphics Corporation"  
`pragma protect key_keyname = "MGC-VERIF-SIM-RSA-2"  
`pragma protect key_method = "rsa"  
`pragma protect key_block encoding = ( enctype = "base64" )  
`pragma protect data_method = "aes128-cbc"
```

You can use the *-wholefile* argument to force the command to encrypt an entire file. This switch ignores existing pragmas within the input file. If you specify the *-data_method* argument on the command line, the command uses it rather than the *data_method* pragma defined in the code above (aes128-cbc).

If you specify the *-allow_veloce* argument with no directives in the input files, the command adds the following:

```
`pragma protect key_keyowner = "Mentor Graphics Corporation"  
`pragma protect key_keyname = "MGC-VELOCE-RSA"  
`pragma protect key_method = "rsa"  
`pragma protect key_block encoding = ( enctype = "base64" )
```

The vencrypt command must be followed by a compile command – such as [vlog](#) – for the design to be compiled.

Autoprotect Mode

The vencrypt command contains a number of arguments for automatic encryption of Verilog-only files. Note the following about automatic encryption:

- Automatic encryption does not require any embedded protect pragmas, or the specification of any encryption key.
- In autoprotect mode, the command ignores any embedded protect directives.
- Encryption is based on modules (text between begin and end statements). However, there is one option to encrypt an entire file regardless of modules.
- Various arguments enable you to control what is visible within a module.
- The command uses V0 ModelSim specific encryption. This is an older encryption algorithm.
- An argument enables PLI users to access encrypted modules.

You can use only one of the "autoprotect" arguments at a time. You cannot use autoprotect mode with the following arguments:

- allow_veloce
- wholefile
- data_method
- common
- toolblock
- keyring
- header

Arguments

- <filename> [<filename> . . .]
(required) Specifies the name of the Verilog source code file to encrypt. Requires one filename. You can enter multiple filenames as a space-separated list. You can use wildcards. Default encryption pragmas are used, as described above, if no encryption directives are found during processing.
- -allow_veloce
Allow Veloce emulator to use encrypted outputs.

- **-autotprotect**
Encrypts the text that appears between a Verilog "module <name>" and "endmodule" statements; module names remain user visible. Encrypts the parameter and port lists. Does not encrypt text that is not contained within a Verilog module.
- **-autotprotect_all**
Encrypts the entire file in one block of encrypted code, fully protecting every design unit, including code outside any Verilog module.
- **-auto2protect**
Encrypts an entire module, except for the port list. The parameter list is encrypted. (Does not encrypt any `define and `include statements inside the module.)
- **-auto3protect**
Encrypts an entire Verilog module, except for the parameter and port list. (Does not encrypt `define and `include statements inside the module.)
- **-auto5protect**
Encrypts in the same way as "-auto3protect", and also encrypts content between `ifdef and `ifndef construct directives, so that only the enclosing `ifdef/`endif are visible. This autotprotect mode also utilizes the meta comments //unprotect and //endunprotect, and does not encrypt them or the text between them.
- **-pli_unprotected**
Used only in conjunction with the "-auto[235]protect" arguments. The resulting encrypted files follow the results of the given "auto[235]protect" argument, but PLI users can access encrypted data in -pli_unprotected modules. PLI users cannot access modules encrypted without this option.
- **-common <filename>**
Specifies a file containing information to make common to all the tool blocks. Common information includes any directives you intend for the common block, along with author, author_info, and data_method directives. The file can contain begin_commonblock and end_commonblock directives, but these are ignored by the command. The file can also contain single-line comments ("// for Verilog). The command searches the specified file for the location of the "keyring" directory. You can specify only one -common file. If you specify the file, then you must supply at least one -toolblock argument.
- **-d <dirname>**
Specifies where to save encrypted Verilog files. If you do not specify a directory, the current working directory is used.

<dirname> — Specifies the directory to contain the encrypted Verilog or SystemVerilog files. The original file extension (.v for Verilog and .sv for SystemVerilog) is preserved.

- **-data_method=[aes128|aes192|aes256]**

Specifies the symmetric encryption method to use with the random session key when encrypting text in a data_block. The data_block consists of the characters that appear between the begin and end directives in the input file. If you do not specify a -common or -toolblock argument, the command creates an IEEE 1735 version 1 compliant encryption envelope using the Mentor Graphics public encryption key. This command line setting overrides any -data_method directives appearing in the input file.

 aes128 — AES key size of 128. This is the default.

 aes192 — AES key size of 192.

 aes256 — AES key size of 256.

- **-e <extension>**

(optional) Specifies a filename extension.

 <extension> — Any alpha-numeric string.

- **-f <cmdfile>**

(optional) Specifies a file containing command line arguments. Allows you to reuse complex arguments without retyping. Allows nesting of -f options.

Refer to “[Argument Files](#)” on page 37 for more information.

 <cmdfile> — Specifies the name of a file containing command line arguments.

- **-hea <headerfile>**

(optional) Concatenates header information into all design files listed with <filename>. Enables you to easily add the same `pragma protect information to a large number of files. Allows you to pass files to the vencrypt utility that do not contain the `pragma protect information about how to encrypt the file.

 <headerfile> — Specifies an existing file.

- **-keyring <directory>**

Specifies the directory containing the common and toolblock files. If you do not specify the directory, the command assumes the directory is underneath the top of the installation directory <top_of_installation_dir>/keyring. This option is useful only if using the -toolblock argument and -common.

- **-logfile <logfile> | -l <logfile>**

(optional) Redirects log output to the file designated by <logfile>.

 <logfile> — Specifies a file for saving output.

- **-o <outputfile>**

(optional) Combines all encrypted output into a single file.

 <outputfile> — 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).
Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts 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 — Has no effect and is ignored.

Modes

Modes can be set for a specific feature, or set 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 kilobyte 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 use -stats=perf+verbose to display information about the operating system and host machine on which the executable is run. You can use vencrypt -quiet to disable all default or user-specified -stats features.

- **toolblock <keyfile>[,<rightsfile>]**

Specifies the root name of a file that contains directives intended for a tool block.

keyfile — The root name of the file. Generally, it is the name of a key-key (key_keyname) contained within the file. This file is in the "keyring" directory, and must include either a ".deprecated" or ".active" suffix to the base root name. The command does not use the suffix as part of the specified filename.

rightsfile — Optionally, a "rights file" name may appear after the keyfile name; This file contains control directives for this particular tool block. Control directives are used to define rights. The rights file must have a ".rights" suffix.

- **-wholefile**

This switch ignores any `pragma protect directives that appear in the input file. The command encrypts all the specified files, referencing the -common and -toolbox arguments to determine how to encrypt the files. If you do not specify the -common or -toolblock arguments, then the command creates a version 1 (V1) encryption envelope, with only one key block: the default key for Mentor Graphics Corporation.

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 \[Questa SIM User's Manual\]](#)

[vhencrypt](#)

verror

Returns a detailed description about a message number or a list of messages related to a specified portion of the product.

Syntax

```
verror [-fmt | -full] <msgNum> [,]...
verror [-fmt | -full] [-kind <tool>] -all
verror [-kind <tool>] {-pedanticerrors | -permissive | -suppressibleerrors}
```

Arguments

- **-fmt | -full**

(optional) Specifies the type and amount of information to return.

-fmt

Returns the format string used in the message.

-full

Returns the format string and complete text associated with the 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 | -suppressibleerrors}**

(optional) Specifies filtering for messages according to the message type.

<tool>

Any of the values allowed for the -kind argument.

-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`.

-suppressibleerrors

Display messages that you can suppress from the command line or *modelsim.ini* file.

- **<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 use either a comma-separated or a space-separated list to specify `<msgNum>` any number of times for one `verror` command.

Optionally, you can specify the toolname prior to the message number, in the same way 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 can enter:

```
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

Takes a compiled Verilog module from a library and writes its equivalent VHDL component declaration to standard output.

Syntax

```
vgencomp [-lib <library_name>] [-b] [-bool] [-modelsimini <path/modelsim.ini>] [-s] [-v] [-work <name>] <module_name>
```

Description

Optional switches allow you to generate bit or vl_logic port types. Default is to generate std_logic port types.

Arguments

- **-lib <library_name>**
(optional) Specifies the working library. 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.
- **-bool**
(optional) Causes vgencomp to generate boolean port types.
- **-modelsimini <path/modelsim.ini>**
(optional) Loads an alternate initialization file to replace 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 name. On Windows systems the path separator is 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.
- **-work <name>**
(optional) Specifies the name of the work library. Default is the library containing the module.
- **<module_name>**
(required) Specifies the name of the Verilog module to access.

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

Encrypts VHDL code contained within encryption envelopes.

Syntax

```
vhencrypt <filename> [<filename> . . .] [-allow_veloce] [-d <dirname>] [-e <extension>]  
[ -f <cmdfile>] [ -hea <headerfile>] [ -logfile <logfile> | -l <logfile>] [ -o <outputfile>]  
[ -p <prefix>] [ -quiet] [ -stats [=+ | -]<feature>[,[+ | -]<mode>]]
```

Description

Upon execution, vhencrypt changes the *.vhd* filename extension to *.vhdp*, and the *.vhdl* filename extension to *.vhdlp*.

The vhencrypt utility does not produce an output file if it does not find any encryption directives.

Since the code is not compiled before encryption, dependent packages and design units do not have to exist before encryption. You must follow the vhencrypt command with a compile command – such as vcom – for the design to be compiled.

Arguments

- <filename> [<filename> . . .]
(required) Specifies the name of the VHDL source code file to encrypt. One filename is required. Enter multiple filenames as a space-separated list. Allows wildcards.
- -allow_veloce
(optional) Allow Veloce emulator to use encrypted outputs.
- -d <dirname>
(optional) Specifies where to save encrypted VHDL files. Defaults to the current working directory, if you do not specify a directory.
 <dirname> — Specifies the directory to contain the encrypted VHDL files. Preserves the original file extension (*.vhd* or *.vhdl*).
- -e <extension>
(optional) Specifies a filename extension to apply to the encrypted file.
 <extension> — Any alpha-numeric string.
- -f <cmdfile>
(optional) Specifies a file with more command line arguments. Allows reuse of complex arguments without retying. Allows nesting of -f options.

Refer to “[Argument Files](#)” on page 37 for more information.

 <cmdfile> — Specifies the name of a file containing command line arguments.

- **-hea <headerfile>**
(optional) Concatenates header information into all design files listed with <filename>. Enables you to pass vhencrypt a large number of files that do not contain the encryption information in the form of the `protect compiler directives. Enables you to avoid having to edit hundreds of files to add the same encryption information.
 <headerfile> — Specifies an existing file.
- **-logfile <logfile> | -l <logfile>**
(optional) Redirects log output to the file designated by <logfile>.
 <logfile> — Specifies a file for saving output.
- **-o <outputfile>**
(optional) Combines all encrypted output into a single file.
 <outputfile> — 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).
Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the default settings "cmd,msg".

Features

all — Display all statistics features (cmd, msg, perf). Mutually exclusive with the **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 the **all** option. When specified in a string with other options, none is applied first.

perf — Display time and memory performance statistics.

time — Has no effect and is ignored.

Modes

Modes can be set for a specific feature, or set globally for all features. To add or subtract a mode for a specific feature, use 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 kilobyte 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 use -stats=perf+verbose to display information about the operating system and host machine on which the executable is run. You can use vhencrypt -quiet to disable 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 \[Questa SIM User's Manual\]](#)

[vencrypt](#)

view

Opens the specified window or, if specified without arguments, returns a list of all open windows in the current layout.

Syntax

```
view <window_type>... [-aliases] [-names] [-new] [-title {New Window Title}]  
[-undock {[ -icon] [-height <n>] [-width <n>] [-x <n>] [-y <n>]}] | -dock]
```

Description

When you use the -new argument with the view command, Questa SIM creates an additional instance of the specified window type and makes it the active window for that type. If multiple instances of a window exist, the view command changes the active window of that type to the specified window.

Window names are generated as follows:

- The first window (automatically generated without using -new) has the same name as the specified window type. For example, the view wave command creates a wave window named "wave."
- Additional window names, created by using the -new argument, append an integer to the window type, starting with 1. For example, using view wave command a second time automatically generates the window name "wave1." Using the command again generates "wave2," then "wave3," and so on.
- By default, the designation “- Default” is appended to the name of the currently active window.
- You can use rename with the -title argument to rename existing windows, or create a name for a new window using -new and -title together.

To remove a window, use the [noview](#) command.

If you do not specify a window name, any view command options apply to the currently open windows. Refer to examples for additional details.

Arguments

- <window_type>...

(required) Specifies the window type to view. You do not need to enter the full type name (see the examples below); accepts implicit wildcards; accepts multiple window types.

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 can also specify which window(s) to view when multiple instances of a 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. You must enclose strings that contain spaces in curly braces or double quotes (" "), 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.
view wave -new mysig

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.

virtual count

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.

Accepts unique abbreviations.

- [-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 \[Questa SIM User's Manual\]](#)

virtual define

Prints the definition of the virtual signals, functions, or regions to the transcript, 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) Prints only a subset of virtuals to the transcript.

`implicits` — Virtual signals created internally by the tool.

`explicits` — Virtual signals explicitly created by a user, such as with the `virtual signal` command.

Accepts unique abbreviations.

- `<pathname>`

(required) Specifies the path to the virtual(s) for which you want definitions. Allows wildcards.

Examples

- Show the definitions of all the virtuals you have explicitly created.

`virtual define -kind explicits *`

Related Topics

[virtual describe](#)

[virtual show](#)

[Virtual Objects \[Questa SIM User's Manual\]](#)

virtual delete

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.

Accepts unique abbreviations.

- `<pathname>`

(required) Specifies the path to the virtual(s) to delete. Allows wildcards.

Examples

- Delete all of the virtuals you have explicitly created.

`virtual delete -kind explicits *`

Related Topics

[virtual signal](#)

[virtual function](#)

[Virtual Objects \[Questa SIM User's Manual\]](#)

virtual describe

Prints a complete description of the data type of one or more virtual signals to the transcript.
Similar to the describe command.

Syntax

`virtual describe [-kind {implicits | explicits}] <pathname>`

Arguments

- `-kind {implicits | explicits}`

(optional) Prints only a subset of virtuals to the transcript.

`implicits` — virtual signals created internally by the product.

`explicits` — virtual signals explicitly created by a user, such as with the `virtual signal` command.

Accepts unique abbreviations.

- `<pathname>`

(required) Specifies the path to the virtual(s) for which you want descriptions. Allows wildcards.

Examples

- Describe the data type of all virtuals you have explicitly created.

`virtual describe -kind explicits *`

Related Topics

[virtual define](#)

[virtual show](#)

[Virtual Objects \[Questa SIM User's Manual\]](#)

virtual expand

Prints a list of all of the non-virtual objects contained in the specified virtual signal(s) to the transcript.

Syntax

`virtual expand [-base] <pathname> ...`

Arguments

- `-base`

(optional) Outputs the root signal parent in place of a subelement. For example, the result of the following command:

```
vcd add [virtual expand -base myVirtualSignal]
```

is:

```
vcd add signala signalb signalc
```

- `<pathname>`

(required) Specifies the path to the signals and virtual signals to expand. Allows wildcards, and you can specify any number of paths.

Description

The virtual expand command enables you to create a list of arguments for a command that does not accept or understand virtual signals.

Examples

- Add the elements of a virtual signal to the VCD file.

In the Tcl language, the square brackets specify to execute the enclosed command first ("virtual expand ..."), then substitute the result into the surrounding command.

```
vcd add [virtual expand myVirtualSignal]
```

If *myVirtualSignal* is a concatenation of *signal_a*, *signal_b.rec1* and *signal_c(5 downto 3)*, the resulting command after substitution is:

```
vcd add signala signalb.rec1 {signalc(5 downto 3)}
```

The slice of *signal_c* is enclosed in curly braces, because it contains spaces.

Related Topics

[virtual signal](#)

[Virtual Objects \[Questa SIM User's Manual\]](#)

virtual function

Creates a new signal, used by the GUI (not the kernel), that consists of logical operations on existing signals and simulation time.

Syntax

```
virtual function [-env <path>] [-install <path>] [-delay <time> <unit>] {<expressionString>}  
    <name>
```

Description

The virtual function command cannot handle bit selects and slices of Verilog registers. Refer to “[Syntax and Conventions](#)” on page 19 for more details on syntax.

If a virtual function references more than a single scalar signal, it displays as an expandable object in the Wave and Objects windows. The children correspond to the inputs of the virtual function. You can expand the function in the Wave window to compare the values of the input waveforms.

You can also use virtual functions to gate the List window display.

Note

 The virtual function and virtual signal commands are interchangeable. The product keeps track of whether you have created a signal or a function with the commands, and maintains 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.

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 do not 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 you do not specify -install, 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 is automatically placed in region virtuals:/ Functions.
 <path> — Specifies a relative path to the signal(s). On Windows systems the path separator is a forward slash (/).

- **-delay <time> <unit>**
(optional) Specifies a value by which to delay the virtual function. 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. Defaults to the current simulation units, if you do not specify <unit>.
 - <unit> — (optional) Specifies a unit of time. 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](#)” on page 42.
- **<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 you use VHDL extended identifier notation, <name> needs to be enclosed in double quotes (" ") or 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 install it as */chip/rega_slv*.
virtual function -install /chip { (std_logic_vector) chip.vlog.reg } 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] == 0xfb } 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*. You can add the virtual function *siga_diff* 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 does not 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 \[Questa SIM User's Manual\]](#)

[virtual nohide](#)

[virtual nolog](#)

[virtual region](#)

[virtual save](#)

[virtual show](#)

[virtual signal](#)

[virtual type](#)

virtual hide

Hides the specified real or virtual signals from display in the Objects window.

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.

Accepts unique abbreviations.

- `-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 is a forward slash (/).

- `<pattern>`

(required) Indicates which signal names to use to find the signals to hide. Allows wildcards, and you can specify any number of names or patterns.

Description

You can use the `virtual hide` command to replace an expanded bus with a user-defined bus. Use the `virtual nohide` command to make the signals reappear.

Related Topics

[virtual nohide](#)

[Virtual Objects \[Questa SIM User's Manual\]](#)

virtual log

Causes the kernel to log the simulation-mode dependent signals of the specified virtual signals.

Note

 When you use wildcard patterns to specify signals to log, normal signals that match the patterns are also logged, unless you include the -only option. You can unlog the signals with the virtual noglog 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.
- Accepts unique abbreviations.
- **-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 is a forward slash (/).
- **-recursive**
(optional) Specifies that the scope of the search is to descend recursively into subregions. Omitting -recursive limits the search to the selected region.
- **-only**
(optional) Specifies to log only virtual signals (as opposed to all signals) found by a *<pattern>* containing a wildcard.
- **-in**
(optional) Specifies that the kernel is to log data for ports of mode IN whose names match the specification.
- **-out**
(optional) Specifies that the kernel is to log data for ports of mode OUT whose names match the specification.

- **-inout**
(optional) Specifies that the kernel is to log data for ports of mode INOUT whose names match the specification.
- **-internal**
(optional) Specifies that the kernel is to log data for internal (non-port) objects whose names match the specification.
- **-ports**
(optional) Specifies that the kernel is to log data for all ports.
- **<pattern>**
(required) Specifies the signal names or wildcard patterns to use to find the signals to log.
You can specify any number of names or wildcard patterns.

Related Topics

[Virtual Objects \[Questa SIM User's Manual\]](#)

[virtual nolog](#)

virtual nohide

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.

Accepts unique abbreviations.

- `-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 is a forward slash (/).

- `<pattern>`

(required) Indicates which signal names or wildcard patterns to use in finding the signals to hide. Allows wildcards, and you can specify any number of names or patterns.

Related Topics

[virtual hide](#)

[Virtual Objects \[Questa SIM User's Manual\]](#)

virtual nolog

Reverses the effect of a virtual log command, causing the simulation-dependent signals of the specified virtual signals to be excluded ("unlogged") by the kernel.

Note

 If you use wildcard patterns to specify signals, the command also unlogs any normal signals it finds, unless you include the **-only** option.

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.

Accepts unique abbreviations.

- **-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 is a forward slash (/).

- **-recursive**

(optional) Specifies that the scope of the search is to descend recursively into subregions. Omitting **-recursive** limits the search to the selected region.

- **-only**

(optional) Specifies to unlog only virtual signals (as opposed to all signals) found by a **<pattern>** containing a wildcard .

- **-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 the signal names or wildcard patterns to use in finding the signals to unlog. |Allows wildcards, and you can specify any number of names or patterns.

Related Topics

[virtual log](#)

[Virtual Objects \[Questa SIM User's Manual\]](#)

virtual region

Creates a new user-defined design hierarchy region.

Note

 You cannot use virtual regions 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 for the new region.

Related Topics

[virtual function](#)

[virtual signal](#)

[Virtual Objects \[Questa SIM User's Manual\]](#)

virtual save

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.

Accepts unique abbreviations.

- -append

(optional) Specifies to save only virtuals that are not already saved or were not read in from a macro file. Appends these unsaved virtuals to the specified or default file.

- <filename>

(optional) The name of the file containing the definitions. If you do not specify <filename>, the default virtual filename (virtuals.do) is used. You can specify a different default in the *pref.tcl* file.

Related Topics

[virtual count](#)

[Virtual Objects \[Questa SIM User's Manual\]](#)

virtual show

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.

Accepts unique abbreviations.

Related Topics

[virtual define](#)

[virtual describe](#)

[Virtual Objects \[Questa SIM User's Manual\]](#)

virtual signal

Creates a new signal, known only by the GUI (not the kernel), that consists of concatenations of the signals and sub-elements specified in <expressionString>.

Syntax

```
virtual signal [-env <path>] [-install <path>] [-delay <time> <unit>] {<expressionString>}  
          <name>
```

Description

The virtual signal command cannot handle bit selects and slices of Verilog registers. Please see “[Concatenation of Signals or Subelements](#)” on page 49 for more details on syntax.

Note

 The virtual function and virtual signal commands are interchangeable. The GUI keeps track of whether you have created a signal or a function with the commands, and maintains 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.

Arguments

- **-env <path>**
(optional) Specifies a hierarchical context in <expressionString> so that you do not have to enter a full path for each signal name.

 <path> — Specifies a relative path to the signal(s). On Windows systems the path separator is a forward slash (/).
- **-install <path>**
(optional) Causes the newly-created signal to become a child of the specified region. If you do not specify -install, 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 is automatically placed in region virtuals:/Signals.

 <path> — Specifies a relative path to the signal(s). On Windows systems the path separator is a forward slash (/).
- **-delay <time> <unit>**
(optional) Specifies the value by which the virtual function is 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. If you do not specify <unit>, the default is the current simulation units.

 <unit> — (optional) A suffix specifying a unit of time. 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](#)” on page 42.

- <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 enclosed in double quotes ("") or 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 of Signals or Subelements](#)” on page 49 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 add](#) [toggle add](#)

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 \[Questa SIM User's Manual\]](#)

virtual type

Creates a new enumerated type known only by the GUI, not the kernel. Converts signal values to character strings.

Syntax

virtual type [-delete <name>](#) | {[<list_of_strings>](#)} <name>

Description

The virtual type 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.

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. You can express values in decimal or based notation and can include "don't-cares" (see examples below). Supports three based notation styles: Verilog, VHDL, and C-language. Interprets values without regard to the size of the bus to be mapped. Supports bus widths up to 64 bits.

Strings that contain spaces must be enclosed in quotation marks (""). Strings that contain special characters, such as square brackets, curly braces, backslashes, and so on, must be enclosed in 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 enclosed in double quotes ("") or curly braces ({}).

Examples

- Use positional notation to associate 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 \[Questa SIM User's Manual\]](#)

vlib

Creates a design library.

Syntax

`vlib -help`

```
vlib [-short | -dos | -long | -unix] [-dirpath <pathname>] [-format { 1 | 3 | 4 }] [-type {directory | archive | flat}] [{-lock | -unlock} <design_unit>] [-locklib | -unlocklib] [-unnamed_designs <value>] [-compress | -nocompress] <library_name>
```

Description

You must use vlib, rather than operating system commands, to create valid Questa SIM library directories or index files. If a specified library already exists, the vlib command exits with a warning message, without touching the library.

Arguments

- `-compress | -nocompress`

(optional) Defines whether to store some compiled results in the library in a compressed form.

-compress — Compression occurs, producing smaller libraries. Compression can slow down subsequent executions of the vopt command.

-nocompress — (default) Library compression does not occur.

- `-dirpath <pathname>`

(optional) Specifies the path to a working directory, which is stored in the library in order to override the current working directory. This enables you hide the directory path information.

Caution

 Use of this argument is not recommended.

If you override the current working directory with -dirpath, the Questa SIM user interface will be unable to find the source files when you select something in the design and ask to see the declaration.

- `-dos`

(optional) Specifies that library subdirectories have names that are compatible with DOS. Not recommended if you use the `vmake` utility.

On by default for ModelSim PE.

- `-format { 1 | 3 | 4 }`

(optional) Prepares a library for conversion to compatibility with a previous release, by altering the `_info` file.

1 — Converts a library to be compatible with the 6.2 series and earlier.

3 — Converts a library to be compatible with the 6.3 series and newer.

4 — Converts 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. These switches do no affect file permissions.
- **-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. These switches do not affect file permissions.

Refer to “[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 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 to allow subdirectories in a library to have long file names that are NOT compatible with DOS.
On by default for Questa.
- **-unnamed_designs <value>**
(optional) Specifies the number of unnamed, optimized versions of a design for the [vopt](#) command to save in the library. Once **<value>** is reached, **vopt** deletes the oldest unnamed,

optimized version. The default maximum number of “unnamed” designs (“_opt[number]”) is 3.

Note

 The unnamed optimized designs limit can be exceeded if multiple concurrent vsim sessions are run with the same 'work' library

- <library_name>
(required) Specifies the pathname of the library to create.

Examples

- Create the design library *design*. You can define a logical name for the library with the **vmap** command, or by adding a line to the library section of the *modelsim.ini* file in the same directory.

vlib design

- Creates the design library *uut* and specifies that any design units compiled into the library are created as archives.

vlib -type archive uut

vlog

Compiles Verilog source code and SystemVerilog extensions into either a specified working library or to the default work library. Accepts compressed SystemVerilog source files (those compressed with zlib).

Syntax

```
vlog [options] <filename> [<filename> ...]

[options]:
[-32 | -64] [-93]
[-addpragmaprefix <prefix>] [-ams]
[-ccflags <"compileopts">] [-compat] [-compile_uselibs[=<directory_name>]] [-constimedassert | -noconstimedassert] [-convertallparams] [+cover[=<spec>]] [-covercebi] [-covercells] [-coverdeglitch {"<n> <time_unit>"}] [-coverenhanced] [-coverexcludedefault] [-coverfec] [-coveropt <opt_level>] [-coverreportcancelled] [-cppinstall <[gcc|g++] version>] [-cpppath <filename>] [-createlib[=compress]] [-cuname <package_name>] [-cuautoname[=file | du]] [-define <macro_name>[=<macro_text>]] [+define+<macro_name>[=<macro_text>]] [-deglitchalways | -nodeglitchalways] [+delay_mode_distributed] [+delay_mode_path] [+delay_mode_unit] [+delay_mode_zero] [-dirpath <pathname>] [-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>] [-feceffort {1 | 2 | 3}] [-force_refresh <design_unit>] [-fsmdebug] [-fsmimplicittrans] [-fsmmultitrans] [-nofsmresettrans] [-fsmssamestatetrans] [-fsmsingle] [-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}]] [+initwire[+{0 | 1 | X | Z}]] [-isymfile] [+iterevaluation] [-L <libname>] [-Lf <libname>] [+libcell | +nolibcell] [+libext+<suffix>] [-libmap <pathname>] [-libverbose=libmap] [-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>] [-msglimit {error | warning | all[,-<msgNumber>,...] | none[,+<msgNumber>,...] | <msgNumber>,[msgNumber,...]}] [-msglimitcount <limit_value> -msglimit [all,|none,] [-+<msgNumber>,[,-+<msgNumber>,...]]]
```

```

[-nocoverfcsingleterm] [-nocovershort] [-nocreatelib] [-nodbgsym]
[-nodebug[=ports | =pli | =ports+pli]] [-noexcludeternary <design_unit>]
[-noForceUnsignedToVhdlInteger] [-nologo] [-nooverrideundef] [+nopathpulse] [-nopsl]
[+nospecify] [-note <msg_number>[,<msg_number>,...]] [+notimingchecks] [-novtblfixup]
[+nowarn<CODE>] [-nowarn <category_number>] [-nosparse]
[+num_opt_cell_conds+<value>]

[-optionset <optionset_name>] [-outf <filename>] [-override_precision] [-
override_timescale[=][ ]<time_unit> / <time_precision>] [-O0 | -O1 | -O4 | -O5]
[+pathpulse] [-pedanticerrors] [-permissive] [-permit_defunct_sv] [-
printfilenames[=<filename>]] [-proftick=<integer>] [-[w]prof=<filename>]
[+protect[=<filename>]] [-pslext] [-pslfile <filename>]

[-quiet]

[-R [<simargs>]] [-refresh]

[-s] [-scdpheader <filename>] [-sfcu] [-skipprotected] [-skipprotectedmodule] [-
skipsynthoffregion] [-smartdbgsym] [-source] [-stats [=+ -]<feature>[,[+ -]<mode>]] [-
suppress {<msgNumber> | <msgGroup>} [,<msg_number> | <msgGroup>] ...] [-sv]
[-svext=[+|-]<extension>[,[+|-]<extension>]...] [-svfilesuffix=<extension>[,<extension>...]
<filename>] [-svinputport=net | var | compat | relaxed] [-svpkgcasesens] [-sv05compat] [-
sv09compat] [-sv12compat]

[-timescale[=][ ]<time_units>/<time_precision>] [-togglecountlimit <int>] [-togglewidthlimit
<int>] [-toggleportsonly] [+typdelays]

[-u]

[-v <library_file>] [-version] [-vlog01compat] [-vlog95compat] [-vmake] [-vv]
[-warning <msg_number>[,<msg_number>,...]] [-warning error] [-warnrbw] [-work
<library_name>] [-writetoplevels <fileName>]

[-y <library_directory>]

```

Description

You can invoke the vlog command from Questa SIM or from the operating system command prompt. You can also invoke it during simulation.

Compiled libraries are major-version dependent. When moving between major versions, you must use the vlog -refresh argument to refresh compiled libraries. 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 vlog command default to treating each Verilog design file listed on the command line as a separate compilation unit. To treat multiple files listed in a single command line as a single compilation unit, use either the vlog -mfcu argument or the

MultiFileCompilationUnit *modelsim.ini* file variable. Refer to [MultiFileCompilationUnit](#) in the User's Manual.

Arguments

- **-32 | -64**

Specifies whether vlog uses the 32- or 64-bit executable. -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 specify these options only 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 the case of Verilog identifiers that contain uppercase letters.

- **-addpragmaprefix <prefix>**

(optional) Enables recognition of pragmas with a user-specified prefix. If you do not specify this argument, pragmas are treated as comments.

All regular synthesis and coverage pragmas are honored.

<prefix> — Specifies a user-defined string. The default is no string, indicated by quotation marks (" ").

You can also use the AddPragmaPrefix variable in the vlog section of the *modelsim.ini* file to set the prefix. Refer to [AddPragmaPrefix](#) in the User's Manual.

- **-ams**

(optional) Enables use of wreal as a data type for a Verilog net. A wreal net represents a real-valued physical connection between structural entities, that does not store its value. A wreal net can be used for real-valued nets driven by a single driver, such as a continuous assignment. If no driver is connected to a wreal net, its value is real zero (0.0). Whereas digital nets have an initial value of 'Z', wreal nets have an initial value of 0.0.

- **-assertdebug**

(optional) Enables you to debug SVA/PSL objects when used with vsim -assertdebug.

- **-ccflags <"compileopts">**

Specifies 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) Compile with the -Wreturn-type, -Wimplicit, -Wuninitialized, and -Wmissing-declarations gcc options.

off — Compile without any warning options.

verbose —Compile with the -Wall gcc option.

strict — Compile with the -Werror gcc option.

- **-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 enables you to ignore them. This option does not account for all event order discrepancies, and using it may degrade performance. Refer to “[Event Ordering in Verilog Designs](#)” in the User’s Manual for additional information.

- **-compile_uselibs[=<directory_name>]**

(optional) Locates source files specified in a `uselib directive (Refer to “[Verilog-XL uselib Compiler Directive](#)” in the User’s Manual), compiles those files into automatically created libraries, and updates the *modelsim.ini* file with the logical mappings to the new libraries. If you do not specify a *directory_name*, 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.

- **-constimedassert**

(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 -noconstimedassert switch has been used previously, or if the ShowConstantImmediateAsserts variable in the vlog section of the *modelsim.ini* file is set to 0 (off).

- **-noconstimedassert**

(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 can also set the ShowConstantImmediateAsserts variable in the vlog section of the *modelsim.ini* file to 0 (off).

- **-convertallparams**

(optional) Enables converting parameters that are 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 collection of various coverage statistics 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.

In cases where multiple, potentially competing +cover arguments are applied, the coverage option specified later in the command line overrides an option specified before.

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.
- e — Collect expression statistics. Collects only FEC statistics.
- s — Collect statement statistics.
- t — Collect toggle statistics. Overridden if 'x' is specified elsewhere.
- x — Collect extended toggle statistics (Refer to "[Toggle Coverage](#)" in the User's Manual for details). This takes precedence, if 't' is specified elsewhere.
- f — Collect Finite State Machine statistics.

See the [-coveropt <opt_level>](#) argument for information on overriding the default level of optimization for coverage for a particular compilation run.

- -covercebi
(optional) Enables backward compatibility of if-else branch merging with parent 'else' branch in branch coverage reports. This merging converts a nested verilog if-else ladder with vhdl like if-elsif-else ladder.
- -covercells
(optional) Enables code coverage of modules defined by 'celldesign' and 'endcelldesign' 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. Refer to [CoverCells](#) in the User's Manual.
- -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 final 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 counts 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. Refer to [CoverDeglitchOn](#) and [CoverDeglitchPeriod](#) in the User's Manual.

- **-coverenhanced**

(optional) Enables non-critical functionality, which may change the appearance or content of coverage metrics. This argument has an effect only in letter releases (10.0a, 10.0b, and so on). Major releases (10.0, 10.1, and so on) enable all coverage enhancements present in previous letter release streams by default, and do not require **-coverenhanced**. Bug fixes important to the correctness of coverage numbers are also enabled by default, without **-coverenhanced**. The details of the enhancements **-coverenhanced** enables are present in the product release notes rather than in the Command Reference. For 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. Refer to [CoverFEC](#) in the User's Manual.

- **-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 are 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> 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 can 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.

The default optimization level is 3. You can edit the [CoverOpt](#) variable in the *modelsim.ini* file to change the default. Refer to [CoverOpt](#) in the User's Manual.

- **-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. Refer to [CoverReportCancelled](#) in the User's Manual.

- **-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
```

When -dpicppinstall or -dpicpppath arguments are also present, the order of precedence in determining the compiler path is the following:

- The -dpicppinstall argument is used with vsim
 - The -dpicpppath argument is used with vsim
 - The -cppinstall argument is used with vsim
 - The -cpppath argument is used with vsim
- **-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.

When -dpicppinstall or -dpicpppath arguments are also present, the order of precedence in determining the compiler path is the following:

- The -dpicppinstall argument is used with vsim
 - The -dpicpppath argument is used with vsim
 - The -cppinstall argument is used with vsim
 - The -cpppath argument is used with vsim
- **-createlib[=compress]**

Creates libraries that do not currently exist.

`compress` — Compresses the created libraries

- **-cuname <package_name>**

(optional) Used only in conjunction with -mfcu. The -cuname argument names the compilation unit (*package_name*) being created by vlog. You can then specify the named compilation unit on the vsim command line, along with the <top> design unit. The purpose of doing so is to force elaboration of the 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
```

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. If a module that depends on that compilation unit exists, you do not need to force the elaboration, for it occurs automatically. If you are using qverilog to compile and simulate the design, this binding issue is also automatically handled properly.

- **-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.

Use this option when 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>
```

This argument differs from +define, in that it allows you to use the plus (+) symbol within the macro.

- **+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. It also overrides all `undef directives in the RTL code — that is, any `undef for that macro is ignored. Use the **-nooverrideundef** option for backward compatibility with previous operation. If a macro is defined with the +define command line option and you use the -nooverrideundef option, the `undef is honored for that macro.

- **-deglitchalways | -nodeglitchalways**

Controls the behavior related to zero-delay oscillations among always_comb and always @* combinatorial 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](#)” in the User’s Manual 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.
- -dirpath <pathname>
(optional) Specifies the path to a working directory, which is stored in the library in order to override the current working directory. This allows you hide the directory path information.

Caution



Use of this argument is not recommended.

If you override the current working directory with -dirpath, the Questa SIM user interface will be unable to find the source files when you select something in the design and ask to see the declaration.

- -dpicppinstall <[gcc|g++] version>
(optional) Specifies the version of the GNU compiler supported and distributed by Mentor Graphics to use to compile the DPI exportwrapper.

<[gcc|g++] version> — The version number of the GNU compiler to use. For example:

```
vsim -dpicppinstall 4.5.0
```

This overrides the DpiCppInstall variable in the *modelsim.ini* file.

When -cppinstall or -cpppath arguments are also present, the order of precedence in determining the compiler path is the following:

- The -dpicppinstall argument is used with vsim
- The -dpicpppath argument is used with vsim
- The -cppinstall argument is used with vsim
- The -cpppath argument is used with vsim

-
- **-dpicpppath <pathname>**

(optional) Specifies an explicit path 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. Refer to [DpiCppPath](#) in the User's Manual.

When -cppinstall or -cpppath arguments are also present, the order of precedence in determining the compiler path is the following:

- The -dpicppinstall argument is used with vsim
 - The -dpicpppath argument is used with vsim
 - The -cppinstall argument is used with vsim
 - The -cpppath argument is used with vsim
- **-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 you can then include in C source code for DPI import functions. Refer to “[DPI Use Flow](#)” in the User's Manual 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 with the -v or -y argument.
- <filename> — Specifies a name for the debugging output file. You cannot use wildcards.

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 is sometimes 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. You cannot use wildcards.

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 is sometimes 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 [error](#) and “[Message Severity Level](#)” in the User’s Manual 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. Refer to [ExtendedToggleMode](#) in the User’s Manual.

- **(-F | -file | -f) <filename>**

(optional) -f, -file and -F: each specifies an argument file with more command-line arguments, allowing you to reuse complex argument strings without retyping. Allows nesting of -F, -f and -file commands. 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 “[Argument Files](#)” on page 37 for more information.

- **-fecoeffort {1 | 2 | 3}**

(optional) The recommended option to set the level of effort for FEC. 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 set the level of effort with the [FecEffort](#) 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. Default is to update the work library; 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**

(optional) Allow access to finite state machines for debugging.

- **-fsmimplicittrans**

(optional) Toggles recognition of implied same state transitions. This setting is off by default.

Multiple command arguments are available for FSM management, allowing you to use any combination of the FSM arguments. Refer to [Advanced Command Arguments for FSMs](#) in the User's Manual.

- **-fsmmultitrans**

(optional) Enables detection and reporting of multi-state transitions when used with the +cover=f argument for vlog or [vopt](#) for FSM sequence coverage.

Multiple command arguments are available for FSM management, and you can use any combination of the FSM arguments. Refer to [Advanced Command Arguments for FSMs](#) in the User's Manual.

- **-nofsmresettrans**

(optional) Disables recognition of synchronous or asynchronous reset transitions. On by default.

This setting also excludes reset transitions in coverage results.

Multiple command arguments are available for FSM management, and you can use any combination of the FSM arguments. Refer to [Advanced Command Arguments for FSMs](#) in the User's Manual.

- **-fsmsamestatetrans**

(optional) Enables recognition of transitions to the same state as valid. By default, the compiler does not consider S0->S0 as a valid transition in the following example:

```
If(cst == S0)
  nst = S0
```

- **-fsmsingle**

(optional) Enables recognition of VHDL FSMs where the current state variable is of type std_logic, bit, boolean, or single-bit std_logic_vector/bit_vector and Verilog single-bit FSMs. Off by default.

Multiple command arguments are available for FSM management, and you can use any combination of the FSM arguments. Refer to [Advanced Command Arguments for FSMs](#) in the User's Manual.

- **-fsmverbose[b | t | w]**

(optional) Provides information about FSMs detected, including state reachability analysis. You can specify any or none of the following values:

- 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.

If you do not specify a value, this argument reports all information, as in the following example:

```
# ** 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)
#      -----
```

If you do not specify this argument, you 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 argument detects 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. On by default.

Multiple command arguments are available for FSM management, and you can use any combination of the FSM arguments. Refer to [Advanced Command Arguments for FSMs](#) in the User's Manual.

- **-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:

- a. Compile <filename> into a library with vlog (without -gen_xml).
- b. 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](#)" in the User's Manual for more details.

Note

 Enabling -hazards implicitly enables the -compat argument. As a result, using this argument can affect your simulation results.

- **-ignorepragmaprefix <prefix>**

(optional) Directs vlog to ignore pragmas with the specified prefixname. All affected pragmas are treated as regular comments. Edit the **IgnorePragmaPrefix** *modelsim.ini* variable to set a permanent default. Refer to [IgnorePragmaPrefix](#) in the User's Manual.

<prefix> — Specifies a user defined string.

- **-ignoresvkeywords= <keyword>[,<keyword>]...**

Instructs the complier to ignore the specified SystemVerilog keywords when encountered in a file with a .v extension, and return them as an identifier.

<keyword>[,<keyword>]... — A comma-separated list of SystemVerilog keywords.

- **+incdir+<directory>**

(optional) Specifies directories to search for files included with `include compiler directives. By default, searches the current directory first and then the directories specified by the

+includer options, in the order they appear on the command line. You can specify multiple +includer options and multiple directories, separated by "+", in a single +includer option.

When you compile a file that imports the OVM package shipped with Questa SIM, you do not need to specify +includer+ to the OVM directory. For example, if your .sv file contains:

```
import ovm_pkg::*;
`include "ovm_macros.svh"
```

The vlog command automatically adds the +includer switch for you, issuing the following note:

```
** Note: (vlog-2286) Using implicit
+includer+<install_dir>/<ovm_dir>/src
from import ovm_pkg
```

If you are using an open version of the OVM, you need to explicitly specify that directory with the +includer switch.

A package import of the built-in UVM library adds an implicit +includer entry derived from the 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 is recompiled. However, if you change the compile options, 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 initialize.

If you do not specify this option, vlog initializes fixed-size arrays of all these types, where fixed-size arrays can 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 randomly initializes 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. If you set these memories to the same values that +initmem initialized them to, events are not generated, except for sequential UDPs that are initialized in the NBA region of time 0.

The +initmem 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#

This argument does 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 initialize.

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).

The +initreg option does not initialize notifier registers. To detect that a register is a notifier, timing checks must be present, which means you cannot compile with the +nospecify or +notimingchecks arguments. If you want to remove timing checks but still detect notifier registers, use vsim +notimingchecks or vsim +nospecify, or use `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.

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. If a sequential UDP does not

contain an "initial" statement, and was not compiled with +initreg in effect, the UDPs initial value is 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 map to 0.

If you do not specify the value, initreg randomly initializes 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 are not generated, except for 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#

This argument does not override any variable declaration assignment, such as:

```
reg r = 1'b0
```

- +initwire[+{0 | 1 | X | Z}]

(optional) Initializes unconnected nets in the interface, module, program, or package to the specified value, whether the nets are declared as ports or normal variables. The default is 0 when +initwire is entered with no value. If +initwire is not used, unconnected nets default to Z.

- -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 TranscriptFile or BatchTranscript File *modelsim.ini* variables (refer to [TranscriptFile](#) or [BatchTranscript](#)).

[BatchTranscriptFile](#) in the User’s Manual.) 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](#)” in the User’s Manual for more information.).

Note

 (ModelSim SE and Questa Simulators only) Questa SIM issues a warning if there are any inconsistencies between the library search you specify for vlog -L or vlog -Lf and the search you specify for the -L or -Lf arguments of the vopt command.

- **+libcell | +nolibcell**

+libcell — (optional) Treats all modules found and compiled by a source library search as though they contain a ‘celldesignate compiler directive, and marks 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 ‘celldesignate compiler directive. This argument restores the default library search behavior after you have used the **+libcell | +nolibcell** argument to change it.

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, the compiler searches for a file with the suffix appended to an unresolved name. You can specify only one **+libext** option, but it can 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 in 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.

- **-libverbose=libmap**

(optional)

Displays library map pattern matching information during compilation. Use this argument to troubleshoot problems with matching filename patterns in a library map file. For example, when a resolved module has a choice between two libraries, this argument tells you which one was selected, confirming that your config file worked.

- **-libmap_verbose**

(optional) Displays library map pattern matching information during compilation.

Note

This argument is being deprecated — the recommended method is to use the **-libverbose=libmap** argument instead. However, for compatibility reasons, **-libmap_verbose** will continue to be supported indefinitely, so it is safe to use until further notice.

- **+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.

- **-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 with the Show_Lint variable in the *modelsim.ini* file. Refer to [Show_Lint](#) in the User's Manual.

- **-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. Refer to [AcceptLowerCasePragmaOnly](#) in the User's Manual.

- **-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 are initialized before any subclass properties.

- **+maxdelays**

(optional) Selects maximum delays from the "min:typ:max" expressions. 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. 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, as is 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.

All global declarations present in both compile file and library files specified with the `-v` argument are lumped together in a single `$unit` scope.

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 use the `MultiCompilationUnit` variable in the `modelsim.ini` file to enable this option (without the `=macro` functionality). Refer to [MultiFileCompilationUnit](#) in the User's Manual.

- **-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**, you can include the package 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 name. On Windows systems the path separator is a forward slash (/).

- **-msglimit {error | warning | all[,-<msgNumber>,...] | none[,+<msgNumber>,...] | <msgNumber>,[msgNumber,...]}**

(optional) Limits messages to the default message limit count of five. Use the **-msglimitcount** argument to change the default message limit count. Specify multiple messages as a comma-separated list.

error — Stops the run when the total number of errors reaches the default count.

warning — Stops the run when the total number of warnings reaches the default count.

all — Limits all messages to the default count except those you specifically exclude. To exclude messages from the limit, supply a comma-separated list of message numbers, each preceded by a minus sign (-), for example “all,-<msgNumber1>,<msgNumber2>,...”.

none — Excludes all messages from the default count except those you specifically include. To include messages to limit to the default count, supply a comma-separated list of message numbers, each preceded by a plus sign (+), for example “none,+<msgNumber1>,+<msgNumber2>,...”.

<msgNumber>[,<msgNumber>,...] — Specifies messages to limit to the default count.

Examples:

- Stop the run when the total number of errors reaches the default count:

```
-msglimit error
```

- Stop the run when the total number of warnings reaches the default count:

```
-msglimit warning
```

- Specifically limit messages 2374 and 2385 to the default count:

```
-msglimit 2374,2385
```

- Limit all messages to the default count, except messages 2374 and 2385:

```
-msglimit all,-2374,-2385
```

- Limit only messages 2374 and 2385 to the default count:

```
-msglimit none,+2374,+2385
```

- **-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. Refer to [MsgLimitCount](#) in the User's Manual.

- **-nocoverfecsingleterm**

Disables coverage for expressions that contain multi-bit sub-expressions but whose resulting output is still single bit.

- **-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. Refer to [CoverShortCircuit](#) the User's Manual.

- **-nocreatelib**

(optional) Stops automatic creation of missing work libraries. Overrides the CreateLib modelsim.ini variable. Refer to [CreateLib](#) in the User's Manual.

- **-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 enabling you to view detailed information about design objects at the source level. Two major GUI features that use this database are source window annotation and textual dataflow.

You should specify this switch only if you know that no one else using the library will 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, because 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 can be accessed through the Dataflow or Schematic window or with commands.

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.

- nodebug=ports — Additionally hides the ports for the lower levels of your design. Use this only to compile the lower levels of the design; if you hide the ports of the top level you cannot 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.

This form leaves a "nodebug" module untraversable by PLI.

- nodebug=ports+pli — Specifies to combine the behavior of =ports and =pli.

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.
- **-nooverrideundef**
(optional) Prevents `undefs from being overridden by macros defined with the +define command line option. If a macro is defined with +define command line option, and -nooverrideundef is also passed as a compile option, the `undef is honored for that macro.
- **+nopathpulse**
(optional) Causes PATHPULSE\$ specparam(s) to be ignored in a module’s specify block.
The default is to ignore PATHPULSE\$ specparam(s).

- **-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](#)” in the User’s Manual 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](#) in the User’s Manual for more information.
- **+nospecify**
(optional) Disables specify path delays and timing checks. Causes \$sdf_annotation() to be ignored.
- **-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 [note](#) and “[Message Severity Level](#)” in the User’s Manual 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.
- **-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 the default behavior of treating the type of a this reference 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 will not execute methods of an uninitialized class type).
- **+nowarn<CODE>**
(optional) Disables warning messages in the category specified by <CODE>; those warnings that include the <CODE> name in square brackets in the warning message. For example, +nowarnRDGN disables the following warning message:

```
** Warning: test.v(15) : [RDGN] - Redundant digits in numeric literal.
```
- **-nowarn <category_number>**
(optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message to suppress. You can include multiple -nowarn switches.

Warnings can be disabled for all compiles via the Main window **Compile > Compile Options** menu command, or the *modelsim.ini* file (refer to [modelsim.ini Variables](#) in the User's Manual).

The following table describes the warning message categories:

Table 3-11. 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 cannot 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 “[Optionsets](#)” for more information.
- **-outf <filename>**
(optional) Specifies a file to save the final list of options to, after recursively expanding all -f, -file and -F files.
- **-override_precision**
(optional) Used with the -timescale argument, this argument overrides the precision of `timescale specified in the source code.
- **-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. You can use an equal sign (=) or a space 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:

 $\{1 \mid 10 \mid 100\} \{s \mid ms \mid us \mid ns \mid ps \mid fs\}$
For example, 10 ns.

time_precision — Unit of measurement for rounding delay values before using them 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.

Refer to "[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.

- +pathpulse

(optional) Enables usage of the PATHPULSE\$ specparam in a module's specify block. Has no effect on modules without PATHPULSE\$ specparam(s). The default is to ignore PATHPULSE\$ specparam(s).

- -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 occurs 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 occurs for literals written without the grave accent 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 occurs 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 is not legal.
- Using realtime data types in SystemVerilog assertion is not legal.

- Using an unsized constant in a concatenation if it is the leftmost value in the list is not legal.
- Calling a virtual function in the constructor of the same class is not legal.
- Using integers to define macro names is not legal.

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 the use of reserved keywords ‘config’ and ‘instance’ 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.

- **-printfilenames[=<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>`. Customer Support uses output from these arguments 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 are also inserted into the output file when you add =<filename>. If you do not use =<filename>, all `include files are encrypted into the work directory as individual files, not merged together into one file.

The command line +protect argument will be fully deprecated in a future release. The simulator commands will issue a warning message any time you use the +protect argument.

- **-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](#)” in the User’s Manual for more information.

- **-quiet**

(optional) Disables output of informative messages about processing the design, such as Loading, Compiling, or Optimizing, but not messages about the design itself.

- **-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 you use the -R option, you must specify the log files for vlog and vsim separately. The file you specify before -R will capture the output of the vlog compiler, and the one you specify 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 Questa SIM uses it to combine the compile and simulate phases together, as you may be used to doing with Verilog-XL. This option is provided to ease the transition to Questa SIM. It is not recommended that you use this option regularly, as you will then incur the unnecessary overhead of compiling your design for each simulation run.

- **-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 can use a specific design name with -refresh to regenerate a library image for that design, but you cannot use a file name.

- **-s**
(optional) Instructs the compiler not to load the standard package. Use this argument only when you are compiling the sv_std package.
- **-scdpihandler <filename>**
(optional) Specifies the name of the SystemC DPI function prototype header file automatically generated from the current compilation. Defaults to *sc_dpiheader.h* when you do not provide a filename. Refer to “[SystemC Procedural Interface to SystemVerilog](#)” in the User’s Manual for a more detailed description.
- **-sfcu**
Instructs the compiler to treat all files in a compilation command line as a separate compilation units. This is the default behavior, and is the inverse of the behavior of **-mfcu[=macro]**.
If this file has global declarations, a local \$unit scope is created for a library file passed through with the **-v** argument.
This switch overrides the MultiFileCompilationUnit variable if it is set to "1" in the *modelsim.ini* file. Refer to [MultiFileCompilationUnit](#) in the User’s Manual.
- **-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 number 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. Refer to [SmartDbgSym](#) in the User’s Manual.
- **-source**
(optional) Displays the associated line of source code before each error message generated during compilation. The default is to display only the error message.
- **-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).

Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the settings in the Stats *modelsim.ini* variable.

<feature>

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.

Note

 Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

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 kilobyte 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 {<msgNumber> | <msgGroup>} {[,<msg_number> | <msgGroup>],...]
- (optional) Prevents the specified message(s) from displaying. The <msg_number> is the number preceding the message 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 [suppress](#) and “[Message Severity Level](#)” in the User’s Manual for more information.

<msgNumber>

A specific message number

<msgGroup>

A value identifying a pre-defined group of messages, based on area of functionality. These groups only suppress notes or warnings. The valid arguments are:

All, GroupNote, GroupWarning, GroupFLI, GroupPLI, GroupSDF, GroupVCD, GroupVital, GroupWLF, GroupTCHK, GroupPA, GroupLRM

- **-sv**

(optional) Enables SystemVerilog features and keywords. By default Questa SIM follows the IEEE Std 1364-2005 and ignores SystemVerilog keywords. If a source file has a ".sv" extension, Questa SIM automatically parses SystemVerilog keywords.

- **-svext=[+|-]<extension>[,[+|-]<extension>]...**

(optional) Enables SystemVerilog language extensions through a comma-separated list of options.

[+ | -] — Controls activation of the *extension*. Remember that command arguments take precedence; any settings to this argument override your settings of the SV Extensions *modelsim.ini* variable. Refer to [SvExtensions](#) in the User's Manual.

+ — Activates the *extension*.

- — Deactivates the *extension*.

If you do not specify either a "+" or "-", the command assumes you are activating the specified *extension*.

<extension> —

Note



Specify multiple extensions as a comma-separated list. For example:

`vlog -svext=+feci,-uslt,pae`

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.

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].

arif — Allows the use of refs in `fork-join_any` or `fork-join_none` blocks inside tasks.

aswe — Enables support for the symmetric wild equality operators `=?=` and `!?=`.

atpi — Uses type names as port identifiers. Disabled when compiling with `-pedanticerrors`.

catx — Allows an assignment of a single unsized constant in a concat to be treated as an assignment of 'default:val'.

ctlc — Casts time literals in constraints to the type: time. The LRM dictates that a time literal, such as "10ns" is to be interpreted as a "realtime" type, but this is not always adhered to, for example:

```
class Foo;
  rand time t;
  constraint c1 {
    t < 10ns; // NON-LRM compliant use of 'real' type
  }
endclass
```

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
```

defervda — Causes SystemVerilog variables that have an initializer in the declaration to trigger top-blocking always blocks at time zero.

ddup — (Drive Default Unconnected Port) Reverts behavior to where explicit named unconnected ports are driven by the default value of the port.

dmsbw — Drives the MSB unconnected bits of the wider hiconn (output) or the wider loconn (input) in an otherwise collapsible port connection. With this extension, zero drives unconnected bits; otherwise, they float.

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 in curly braces ({}) in `include string literals and in `include path names. For example, if MYPATH exists in the environment then it will be expanded in the following:

```
`include "$MYPATH/inc.svh"
```

feci — Treats 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 — Iterates 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 — Ignores the DPI task disable protocol check.

ifslvbefr — Allows a solve/before constraint within an IfElse constraint with a constant condition. This is on by default.

mewq — Allows macro substitution inside string literals.

mtdl — Ignores commas and right parentheses occurring within a `" ``` (back tick-double quotation mark back tick-double quotation mark) string that forms a list of macro argument actuals within another macro expansion.

mttd — Reinterprets `" ` (back tick-back tick-double quotation mark) as `" ` (back tick-double quotation mark) within macro text expansion.

ncref — Prevents a ref argument in the new operator of a covergroup to be treated as a constant, unless specified.

omaa — Allows shuffle ordering method on associative arrays.

pae — Automatically export all symbols imported and referenced in a package.

pae1 — Allows the export, using wildcard export only, of package symbols from a subsequent import of a package. These symbols may or may not be referenced in the exporting package.

[+]realrand — Enables randomization of ‘real’ variables and constraints, with the exception of multiply and divide with ‘real’ operands in constant expressions. This extension is enabled by default. Use “-svext=-realrand” to disable this extension at compile time.

Note

 Randomization of ‘real’ variables or constraints requires a SystemVerilog Real Number Modeling (svrnm) license. If the license check fails, ‘real’ number support is disabled.

sas — Enables LRM-compliant @* sensitivity rules.

sccts — Processes string concatenations converting the result to string type.

sceq — Allows string comparison with SystemVerilog case equality operator (==).

scref — Allows you to specify ref arguments in covergroup sample functions without using const.

spsl — (default) Searches 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.

substr1 — Allows one argument in the builtin function substr. A second argument will be treated as the end of the string.

thrdrngshfl —Determines which random number generator the simulator uses for the array manipulation method (array.shuffle). Setting this to +thrdrngshfl uses a RNG of the active thread, while the default, -thrdrngshfl, uses a separate internal RNG.

tzas — Causes an always block to behave as though triggered at time zero, even if none of the variables and nets in the implied sensitivity change value at time zero. This extension runs a top-blocking always @* at time zero, as is done for an always_comb.

ubdic — Allows the use of a variable in a SystemVerilog class before it is defined. For example:

```
class A;
    function func();
        int y = x;           // variable 'x' is used before it is defined
    endfunction
    int x = 1;
endclass
```

If you do not enable this extension, you will receive unresolved reference errors.

udm0 — Expands any undefined macro with the text “1'b0”.

uslt — (default) Promotes 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.

- **-svfilesuffix=<extension>[,<extension>...]** <filename>

Allows specification of filename extensions for SystemVerilog files. Overrides the SVFileSuffixes variable in the *modelsim.ini* file for specified <filename>. Refer to [SVFileSuffixes](#) in the User’s Manual.

- **-svinputport=net | var | compat | relaxed**

(optional) Used in conjunction with -sv (or files with an .sv extension) to determine the default kind for an input port that is declared with a type, but with out the var keyword.

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 the default net type as specified by the ‘default_nettype’ directive.

var — declares the port to be a variable. This value enforces behavior from previous releases, where the port declaration defaults to variable.

compat — declares that only types compatible with net declarations default to wire.

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.

- **-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. If you use a space between arguments, you must enclose <time_units> / <time_precision> 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 \(Verilog\)](#)” in the User’s Manual 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. Refer to [ToggleCountLimit](#) in the User’s Manual.

- **-toggleportsonly**

(optional) Enables ports only for inclusion in toggle coverage numbers; internal signals are not included in coverage metrics. To see coverage of all ports in the design, use the “vopt +acc=p” command — but note that this command turns off inlining, which can result in a significant performance penalty. 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. Refer to [TogglePortsOnly](#) in the User’s Manual.

- **-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. Refer to [ToggleWidthLimit](#) in the User's Manual.

- **+typdelays**

Selects typical delays from the "min:typ:max" expressions. Default. You can specify the same option to the simulator to defer delay selection until simulation time.

- **-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"](#) in the User's Manual 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. You can have multiple -v options. See additional discussion in the examples.

- **-version**

(optional) Returns the version of the compiler as used by the licensing tools.

- **-vlog01compat**

(optional) 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. Refer to [vlog95compat](#) in the User's Manual.

- **-vmake**

Generates a complete record of all command line data and files accessed during the compile of a design. This data is 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.

- **-vv**

Prints to stdout the C/C++ compile and link subprocess command line 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 [warning](#) and "[Message Severity Level](#)" in the User's Manual 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. Can be specified only when compiling the top level modules.

<fileName> — Required. Specifies the name of the file in which to record module names.

- **-y <library_directory>**
(optional) Specifies a source library directory containing definitions for modules, packages, interfaces, and user-defined primitives (UDPs). Typically, this is a directory of source files to scan if the compiled versions do not already exist in a library. Refer to "["Verilog-XL Compatible Compiler Arguments"](#)" in the User's Manual for more information.

After processing all explicit filenames on the vlog command line, the compiler uses the -y option to find and compile any modules that were referenced but not yet defined. Files in this directory are compiled only if the file names match the names of previously unresolved references. You can use multiple -y options. You must 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. A filename is required. You can enter multiple filenames, separated by spaces. You can use wildcards.

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 also restricts the use of PLI functions.

- 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 cannot 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](#)” in the User’s Manual 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

Enables you to use a MAKE program to maintain individual libraries.

Note

 You must invoke this command from the system prompt. If a design is spread across multiple libraries, then each library must have its own makefile and you must build each one separately.

Syntax

```
vmake [-cygdrive] [-du <design_unit_name> ...] [-f <filename>] [-fullsrcpath] [-ignore]
      [<library_name>] [-modelsimini <path/modelsim.ini>]
```

Description

You run vmake on a compiled design library. It operates on multiple source files in the design unit, supporting Verilog include files, as well as Verilog and VHDL PSL vunit files.

By default, the output of vmake is sent to stdout. 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.

A MAKE program is included with Microsoft Visual C/C++, as well as many other program development environments.

Note

 Makefile support for flat libraries is limited because of the lack of per-target file objects. However, the resulting makefile does trigger a build of all design units if any source file for any design unit is newer than the library. Optimized design units in flat libraries are also supported, with more precise dependency tracking.

You run vmake once; then run MAKE to rebuild your design. MAKE recompiles only the design units (and their dependencies) that have changed. 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.

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 to generate a vmake file for only 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](#)” on page 37 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 in directories that are different than the directory 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 no name is specified, then work is assumed.
- **-modelsimini <path/modelsim.ini>**
(optional) Loads an alternate initialization file to replace 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 name. On Windows systems, the path separator is 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
- To use vmake and MAKE on your work library:
C:\MIXEDHDL> vmake >makefile
- To edit an HDL source file within the work library:
C:\MIXEDHDL> make

Your design gets recompiled for you. You can change the design again and re-run MAKE to recompile additional changes.

- To run vmake on libraries other than work:

C:\MIXEDHDL> vmake mylib >mylib.mak

- To rebuild mylib, specify its makefile when you run MAKE:

C:\MIXEDHDL> make -f mylib.mak

vmap

Defines a mapping between a logical library name and a directory by modifying the *modelsim.ini* file.

Syntax

```
vmap [-c | -del <logical_name> ... | <logical_name> [<path>] ]  
      [-modelsimini <path/modelsim.ini>]
```

Description

With no arguments, vmap reads the appropriate *modelsim.ini* file(s) and prints to the transcript the current logical library to physical directory mappings.

Arguments

- **-c**
(optional) Copies the default *modelsim.ini* file from the Questa SIM installation directory to the current directory.
Use this argument only to copy 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 name. On Windows systems the path separator is 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

Performs global optimizations on designs after they have been compiled with vcom or vlog.

Syntax

```
vopt [arguments] <design_unit> -o <name>  
[arguments]:  
[-32 | -64]  
[{+acc | +noacc}[=<spec>][+<selection>[+<recurse_level>.]]] [-adaptive] [-add_seq_delay  
<value>{units}]  
[-cellreport <filename>] [+check<enum>] [-compat] [-constimedassert |  
-noconstimedassert] [-cppinstall <[gcc|g++] version>] [-cpppath <filename>] [{+cover |  
+nocover}[=<spec>][+<selection>[+<recurse_level>.]]] [-covercebi] [-coverclkoptbuiltins |  
-nocoverclkoptbuiltins] [-coverconstruct <comma-separated_mnemonics>]  
[-coverdeglitch {"<n><time_unit>"}] [-coverenhanced] [-coverexcludedefault] [-coverfec]  
[-covermode <options>] [-coveropt <opt_level>] [-coverreportcancelled] [-  
createlib[=compress]]  
[-debug[,viseffort=full | high | low]] [-debugdb] [-debugCellOpt] [-deferSubpgmCheck]  
[-degitchalways | -nodegitchalways] [+delay_mode_distributed] [+delay_mode_path]  
[+delay_mode_unit] [+delay_mode_zero] [-dpiforceheader] [-dpiheader <filename>] [-  
dpilib <libname>]  
[-enablescstdout] [-error <msg_number>[,<msg_number>,...]] [-extendedtogglemode 1|2|3]  
[(-F | -file | -f) <filename>] [-fatal <msg_number>[,<msg_number>,...]] [-feceffort {1 | 2 | 3}]  
[-floatgenerics[+<selection>[+<recurse_level>.]]]  
[-floatparameters[[+<selection>[+<recurse_level>.]] | [+<leafparameter>]]]  
[-fsmimplicittrans] [-fsmmultitrans] [-nofsmresettrans] [-fsmsamestatetrans][-fsmsingle]  
[-fsmverbose [b | t | w]]  
[-g <Name>=<Value> ...] [-G <Name>=<Value> ...] [-gbIso <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>.]])] [+initwire[+{0 | 1 | X | Z}]]  
[-j {0 | 1| <n>}]  
[-keep_delta]
```

```

[-Ldir <pathname> [<pathname> ...]] [-L <libname>] [-Lf <libname>]
[-libertyfiles=<filename> [,<filename>,<filename>,...]] [-libverbose[=prlib]] [-logfile
<filename>] [-lowercaseplpragma] [-lrmclassinit]

[+maxdelays] [-memopt[=n]] [+mindelays] [-mixedansiports] [-modelsimini <path/
modelsim.ini>] [-msglimit {error | warning | all[,-<msgNumber>,...] |
none[,+<msgNumber>,...] | <msgNumber>,[msgNumber,...]}] [-msglimitcount
<limit_value> -msglimit [all,|none,] [-+]<msgNumber>,[,-+]<msgNumber>...]] [-
msglimitcount <limit_value> -msglimit [error|warning]]

[+nocheck<enum>] [-nocoverfecsingleterm] [-nocovershort] [-nocreatelib] [-nodebug[=ports |
=pli | =ports+pli]] [-noexcludeternary] [-nofsmxassign] [+nolibcell] [-nologo] [-noltop]
[+nopathpulse] [-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>] [-override_precision]

[+pathpulse] [-pdu] [-pduignore[=<instpath>]] [-pdusavehierrefs[=<filename>]] [-
pduspec{[+<instancePathName>] | [+<libName>.]<moduleName>} [+facc=<fileName>]]
[-permissive] [-[w]prof=<filename>] [-proftick=<integer>] [-pslext] [-pslfile_vh
<filename>] [-pslfile_vl <filename>]

[-quiet]

[-reporthrefs+ | -reporthrefs] [-rnmautointerconnect]

[-sc22] [-sc_arg <string> ...] [-sclib <library>] [-sdfmin | -sdfty | -sdfmax[@<delayScale>
<instance>=<sdf_filename>] [-sdfmaxerrors <n>] [-seq_udp_delay <value>[<units>]]
[-showsubprograms | -noshowsubprograms] [-source] [+staticchecks[=<args>]] [-
staticchecksfile <filename>] [-staticchecksmdvhdl] [-stats [=+[ | -]<feature>,[,+ | -
]<mode>]] [-suppress {<msgNumber> | <msgGroup>} ,[,<msg_number> | <msgGroup>
,...]] [-svext=[+|-]<extension>,[,+|-]<extension>,...]

[-tab <tabfile>] [-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=resolve|pass|trap|none][,report=fatal|error|warning|info|suppress]
 [,object=<hier_path> | <du>[/<inst>] | <hierlevel>]]

[-xprop_disable=stducheck] [-xprop_enable=[glitchfreeassert] [indexcheck] [optmode]]
```

Using vopt for Power Aware Simulation

The following optional arguments are specific to Power Aware Simulation. For more information, refer to the [Power Aware Simulation User's Manual](#).

```
vopt <design_unit> -o <name>
[-pa_checks=<check_options>] [-pa_checkoption=<value>] [-pa_connectppin=i | a | e | c]
[-pa_corrupt={real | integer} | -pa_nocorrupt={real | integer}] [-pa_coverage [=checks]
[=crosscov] [=crossdontcare] [=implicitportnet] [=iso] [=portpststate] [=powerstate] [=ret]
[=simMode] [=switch] [=undeftrans]] [-pa_crosscoveredepth <integer>] [-pa_db
<path_file_name>] [-pa_dbgfindobj=<filename>] [-pa_defertop]
[-pa_disable=<value>[+<value>]...] [-pa_dumpimdb]
[-pa_dumplibertydb=<database_filename>] [-pa_dumpupf <filename>] [-pa_enable=
<value>[+<value>]...] [-pa_excludedfile <filename>] [-pa_genrpt= [ud] [pa[+b]] [de[+b]]
[cell] [conn] [srcsink] [-pa_gls] [-pa_hiersep <alphanum_character>] [-pa_intcptval0]
[-pa_interactive] [-pa_lib <library_pathname>]
[-pa_libertyfiles=<liberty_filename>[,<liberty_filename>...]]
[-pa_libertyrefresh=<database_filename>] [-pa_libertyNestedCmnts] [-pa_libertyupdate]
[-pa_lsthreshold <real>] [-pa_loadlibertydb=<database_filename>] [-pa_maxpstcol
<integer>][-pa_noinpcorr] [-pa_nopartialontrans] [-pa_opt=nogls][-pa_pgoutsynchchk]
[-pa_prefix <hier_path>] [-pa_prepuffile <filename>]
[-pa_powertop <work_lib.power_module_name>] [-pa_pstcompflags=[p | g | pg | pstids |
useallps]] [-pa_replacetop <string>] [-pa_reportdir <pathname>] [-pa_rslvnetcheck]
[-pa_rtlinfo] [-pa_staticchecksonly] [-pa_tclfile <filename>] [-pa_top <pathname>]
[-pa_upf <filename>] [-pa_upfextensions=<extension>[+<extension>]] [-pa_upflist
<filename>] [-pa_upfsanitychecks] [-pa_upfversion=[1.0 | 2.0 | 2.1 | 3.0]] [-pa_zcorrupt]
```

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>-pa_db
<path_file_name>
```

Description

Use the vopt command before simulation to produce an optimized version of your design in the working directory, using the -o argument to provide a name for the optimized version. You can then run the vsim command directly on that new design unit name.

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 you use vopt on a design that has minimal modifications.

In the course of optimizing a design, vopt removes objects that are not necessary for simulation. For example, vopt removes line numbers, merges processes, may remove some nets and registers, and so on. If you need visibility into your design for debugging purposes, use the +acc argument to conditionally enable visibility for parts of your design.

For quick reference information on vopt arguments, enter vopt -help or vopt -h on the command line.

Syntax conventions

The vopt command makes use of the PathSeparator variable from the *modelsim.ini* file, which enables you to specify the character that Questa SIM recognizes as a path separator (refer to [PathSeparator](#) in the User's Manual). By default, vopt recognizes the forward-slash (/), but you can change this to a period (.) for ease of use with Verilog designs, as shown in the following example:

```
vopt top -o opttop +acc=rn+.top.middle.bottom -G.top.myparam=4
```

All arguments to the vopt command are case-sensitive: -WORK and -work are not equivalent.

Argument overlap with compilation commands

You can also specify many of the vopt arguments described below when you compile with the vcom or vlog commands. These "in-common" arguments are processed by vopt automatically; if you specify these arguments at compile time, vopt will use them during optimization. The first instance of an in-common argument takes priority.

For example, if you specify vcom -O5 or vcom -O5, and then specify vopt -O1, the -O5 argument takes precedence. There is one exception to this behavior: vopt will "OR" any +cover arguments to vlog or vcom (vlog +cover=bc) with any +cover arguments to vopt (vopt +cover=est).

Arguments

- **-32 | -64**

Specifies whether vopt uses the 32- or 64-bit executable. The default is -32.

These arguments apply only to the supported Linux operating systems.

These arguments override the MTI_VCO_MODE environment variable, which applies only to executables from the <install_dir>/bin/ directory. When you run vopt from an <install_dir>/<platform>/ directory, it ignores these arguments.

You can specify these options only on the command line; they are not recognized as part of a file specified 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.

=<spec> — Optionally, one or more of the following characters. The default is to apply the entire set of access specifiers, if you omit <spec>. You can include a non-negative integer with <spec> 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 transcripts only assertion failure messages,

and reports only failure counts in the assertion browser. If a PSL or SystemVerilog construct is being driven by a port signal, vopt may replace that signal name with its higher-level driver. To keep the local port name, you must also specify the +acc "p" option (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. Some user interface commands also require this access to enable you 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 to specific Verilog design objects and/or regions.

<selection> — Can take one of the following forms:

- [<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.
- <instance_path> — A path to an instance.

- <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 (0 to 127) for selection to occur downward from the current level of the specified module or instance. Specifying 128 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
```

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. The following example gives access to all modules whose names begin with “m”:

```
+acc=rn+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
```

The instance can be a scope name as well; 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 finds all instances of the module *mid* and applies +acc=rn to all occurrences of *bottom/myReg*. For example, 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**

(optional) Enable or disable debugging SVA/PSL objects when used with vsim -assertdebug. By default, -assertdebug affects all design units.

- **-adaptive**

(optional) Used with the +cover argument to generate adaptive information for use with adaptive coverage exclusion do file.

- **-add_seq_delay <value>{units}**

(optional) Specifies a constant delay value for all sequential UDPs in the design.

If the existing delay value of sequential UDP is zero, the new delay value is used.

If the existing delay value of sequential UDP is higher, the new delay value is ignored.

If the existing delay value of sequential UDP is lower, the new delay value is replaced.

No other structural delay will be changed with this argument.

- **-cellreport <filename>**

Writes a list of the Verilog modules that qualified for and received gate-level cell optimizations to the specified output file. Does not report optimized cells in a Preoptimized Design Unit (black-boxed region). To determine the cells optimized in a PDU, you need to include the "-cellreport" argument on the vopt command line used in the PDU step.

- **+check<enum>**

(optional) Prevents cell-level optimization on cells with various types of design characteristics, where <enum> determines the type to check, as listed below.

Note

 Optimization on all cells is enabled by default. This argument prevents optimization of cells that model a particular type of behavior. You can specify any one or all of these behaviors to prevent optimization on those cell types. To restore optimization on cells that you have used +check on, use the [+nocheck<enum>](#) argument.

ALL — Applies 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.

INBUF — Prevents optimization of cells with input buffers.

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](#)” in the User’s Manual 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 which version of the Mentor Graphics supported and distributed GNU compiler to use.

<[gcc|g++] version> — The version number of the GNU compiler to use. Use the same compiler version specified on the [sccom](#) command line. For example:

```
sccom -cppinstall 4.5.0
```

When the -cpppath argument is also present, the order of precedence in determining the compiler path is the following:

- The -cppinstall argument is used with vopt
- The -cpppath argument is used with vopt

- `-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.

When the `-cppinstall` argument is also present, the order of precedence in determining the compiler path is the following:

- The `-cppinstall` argument is used with vopt
- The `-cpppath` argument is used with vopt
- `{+cover | +nocover}[=<spec>] [+<selection>[+<recurse_level>|.]]`

(optional) Enables/disables collection of various coverage statistics 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.
 - e — Collect expression statistics. Collects only FEC statistics.
 - s — Collect statement statistics.
 - t — Collect toggle statistics. Overridden if 'x' is specified elsewhere.
 - x — Collect extended toggle statistics (refer to “[Toggle Coverage](#)” in the User’s Manual 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:

- `[<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.
- `<instance_path>` — A path to an instance.

- <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.

Use the **-coveropt <opt_level>** argument to override the default level of optimization for coverage for a particular compilation run.

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*.

In cases where multiple, potentially competing +cover/+nocover arguments are applied, the coverage option specified later in the command line overrides an option specified before.

- **-nocovershort**
(optional) Disables short circuiting of expressions when coverage is enabled. Short circuiting is enabled by default.
- **-coverclkoptbuiltins | -nocoverclkoptbuiltins**
(optional) Enables/Disables clkOpt optimization builtins for code coverage. Used to disable code coverage for VHDL simple builtin flops. You can customize the default behavior with the CoverClkOptBuiltins variable in the *modelsim.ini* file. Refer to [CoverClkOptBuiltins](#) in the User's Manual.
- **-covercebi**
(optional) Enables backward compatibility of if-else branch merging with parent 'else' branch in branch coverage reports. This merging converts a nested verilog if-else ladder with vhdl like if-elsif-else ladder.
- **-coverconstruct <comma-separated_mnemonics>**
Enables you to provide mnemonics that correspond to particular HDL code constructs, which can be instrumented for coverage collection. For example, the coverconstruct "cif" corresponds to the HDL code construct: "condition coverage inside a task or function." The argument can include multiple code construct mnemonics in a comma-separated list. This argument overrides the CoverConstruct *modelsim.ini* variable. Refer to [CoverConstruct](#) in the User's Manual.
- **-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 final execution during that period is added into the coverage data for that process/continuous assignment. If you specify a deglitch period of zero, 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 CoverDeglitch Period variables in the *modelsim.ini* file. Refer to [CoverDeglitchOn](#) and [CoverDeglitchPeriod](#) in the User's Manual.
- **-coverenhanced**
(optional) Enables non-critical functionality that can possibly change the appearance or content of coverage metrics. This argument has an effect only in letter releases (10.0a, 10.0b, and so on). By default, major releases (10.0, 10.1, and so on), enable all coverage

enhancements present in previous letter release streams, making it unnecessary to use -coverenhanced. Also by default, bug fixes important to the accuracy of coverage numbers are always enabled. The product release notes cover details on the release-specific functionality enabled by -coverenhanced. To view 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. Excludes all forms of code coverage collection for statements contained in the OTHERS branch. 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. Excludes all forms of code coverage collection for statements contained in the default case statement.

- **-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. Refer to [CoverFEC](#) in the User's Manual.

- **-covermode <options>**

(optional) Controls the set of cover constructs that are being considered for coverage collection. Options include: full, **default**, set1, set2, and fast. The “default” option is the default value. This argument overrides the CoverMode *modelsim.ini* variable. Refer to [CoverMode](#) in the User's Manual.

- **-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 can 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.

The default optimization level is 3. You can edit the CoverOpt variable in the *modelsim.ini* file to change the default. Refer to [CoverOpt](#) in the User's Manual.

- **-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 functionality with the CoverReportCancelled *modelsim.ini* variable. Refer to [CoverReportCancelled](#) in the User's Manual.

- **-createlib[=compress]**

Creates new libraries.

compress — Compresses the created libraries

- **-debug[,viseffort=full | high | low]**

(optional) Enables increased visibility into VHDL, Verilog, and SystemVerilog designs.

This argument is related to Questa Fast Dumping. You can use the debug information generated with this argument, without the need for the +acc argument, with Mentor Visualizer, and with the following simulation-based TCL-based debugging utilities: VCD dumping, SAIF generation, the Power commands, and the TCL examine command.

-debug

The tool attempts to provide 100% visibility, similar to the setting of **-debug,viseffort=high**.

-debug,viseffort=<value> — (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 — Enables 90% to 100% debug visibility depending on the design, but does not include visibility to floating logic.

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. The simulator reports indexing and length errors in subprograms that are invoked during simulation, 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 @* combinational 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](#)” in the User’s Manual 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 can then be included in C source code for DPI import functions. Refer to “[DPI Use Flow](#)” in the User’s Manual for additional information.

- **-dpilib <libname>**
(optional) Specifies the library that contains DPI exports and object files.
- **-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 is always 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 "["error"](#)" and "["Message Severity Level"](#)" in the User's Manual 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' transitionsEdit the ExtendedToggleMode variable in the *modelsim.ini* file to set a permanent default. Refer to [ExtendedToggleMode](#) in the User's Manual.
- **(-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. You can nest -F, -f and -file commands. You can use 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 "["Argument Files"](#)" on page 37 for more information.
- **-feceffort {1 | 2 | 3}**
(optional) The recommended option to set the level of effort for FEC. 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 [FecEffort](#) 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 "["fatal"](#)" and "["Message Severity Level"](#)" in the User's Manual for more information.

-
- `-floatgenerics[+<selection>[+<recurse_level>].]`

(optional) Does not hold generic values constant during optimization, which enables successful use of the vsim -g and -G arguments. Use of this argument causes finite state machines (FSMs) containing these floating parameters not to be recognized.

Note that the `+floatgenerics` argument is still supported for backward compatibility, but it is recommended that you use `-floatgenerics` instead.

`+<selection>[+<recurse_level>].` — enables access for specific Verilog design objects and/or regions.

`<selection>` — Can take one of the following forms:

- `[<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.
- `<instance_path>` — A path to an instance.
- `<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 for selection to occur downward from the current level of the specified module or instance. This can be any integer between 0 and 127, where a value of 128 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 “[Optimization of Parameters and Generics](#)” in the User’s Manual for more information.

This command is fully equivalent to +floatparameters, therefore you can use them interchangeably.

- +floatparameters[[+<selection>[+<recurse_level>|.]] | [+<leafparameter>]]
(optional) Disables locking down parameter values during optimization, which enables successful use of the vsim -g and -G arguments. Use of this argument can cause FSMs containing these floating parameters not to be recognized.

Note

 The +floatparameters argument is still supported for backward compatibility, though it is recommended that you use -floatparameters instead, with the following exception: if you want to float parameters during optimization without specifying a design object, use vopt +floatparameters.

For the full description of the available values for +floatparameters, refer to [-floatparameters\[\[+<selection>\[+<recurse_level>|.\]\] | \[+<leafparameter>\]\]](#), below.

- -floatparameters[[+<selection>[+<recurse_level>|.]] | [+<leafparameter>]]
(optional) Disables locking down parameter values during optimization, which enables successful use of the vsim -g and -G arguments. Use of this argument can cause FSMs containing these floating parameters not to be recognized.

If you do not specify one of the following values for this argument, Questa SIM does not perform optimization and issues the following warning message:

```
** Warning: (vopt-14001) ignoring vopt option '-floatparameters' as no object is specified.
```

Tip

 The +floatparameters argument is still supported for backward compatibility, but it is recommended that you use -floatparameters instead. However, if you want to float parameters during optimization without specifying a design object, use vopt +floatparameters.

+<selection>[+<recurse_level>|.] — enables access for specific Verilog design objects and/or regions.

<selection> — Can take one of the following forms:

- [<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.
- <instance_path> — A path to an instance.
- <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 “[Optimization of Parameters and Generics](#)” in the User’s Manual for more information.

This command is fully equivalent to `-floatgenerics`, therefore you can use them interchangeably.

- `-fsmimplicittrans`

(optional) Enables recognition of implied same state transitions.

Multiple command arguments are available for FSM management, allowing you to use any combination of the FSM arguments. Refer to “[Advanced Command Arguments for FSMs](#)” in the User’s Manual.

- `-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.

Multiple command arguments are available for FSM management, allowing you to use any combination of the FSM arguments. Refer to “[Advanced Command Arguments for FSMs](#)” in the User’s Manual.

- `-nofsmresettrans`

(optional) Disables recognition of synchronous or asynchronous reset transitions. On by default. This also excludes reset transitions in coverage results.

Multiple command arguments are available for FSM management, allowing you to use any combination of the FSM arguments. Refer to “[Advanced Command Arguments for FSMs](#)” in the User’s Manual.

- `-fsmsamestatetrans`

(optional) Enables recognition of transitions to the same state as valid. By default, the compiler would not consider `S0->S0` as a valid transition in the following example:

```
If (cst == S0)
    nst = S0
```

- `-fsmsingle`

(optional) Enables recognition of VHDL FSMs where the current state variable is of type `std_logic`, `bit`, `boolean`, or single-bit `std_logic_vector/bit_vector` and Verilog single-bit FSMs.

You can use any combination of the multiple command arguments that are available for FSM management. Refer to “[Advanced Command Arguments for FSMs](#)” in the User’s Manual.

- `-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 a value, this argument 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 argument, 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 from 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 "[Optimization of Parameters and Generics](#)" in the User's Manual for additional information.

Limitation: In general, you cannot set generics/parameters of composite type (arrays and records) from the command line, with the exception of string arrays, std_logic vectors, and bit vectors that 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 also overrides generics/parameters that received explicit values in generic maps, in instantiations, or from defparams. This argument is the only way 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 “[Optimization of Parameters and Generics](#)” in the User’s Manual 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 for other shared objects to reference and use. If you specify multiple, comma-separated, shared objects, they are merged internally and then loaded as a single shared object.

If any of the shared objects have circular dependencies you must enclose all the arguments in quotation marks (""), for example:

```
-gblso "lib1.so, lib2.so"
```

You can also specify this argument with the GlobalSharedObjectList variable in the *modelsim.ini* file. Refer to [GlobalSharedObjectList](#) in the User’s Manual.

- **-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](#)” in the User’s Manual 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 optimize only 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) Enables 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 can 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 randomly initializes 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 initialization values 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 are not 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:

- [<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.
- <instance_path> — A path to an instance.

- <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 for selection to occur downward from the current level of the specified module or instance. This can be any integer between 0 and 127, where a value of 128 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 does 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 InitOutCompositeParam variable to 1 in the *modelsim.ini* file. Refer to [InitOutCompositeParam](#) in the User’s Manual.

- **-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, designs 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 InitOutCompositeParam variable to 2 in the *modelsim.ini* file. Refer to [InitOutCompositeParam](#) in the User’s Manual.

- **+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 was not compiled with +initreg in effect, the UDPs initial value is 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 are not generated, with the exception 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:

- [<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.
- <instance_path> — A path to an instance.
- <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 127, where a value of 128 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:

- a. Design unit or instance specific options specified during `vopt` — for example, `vopt ... +initreg+1+top.foo`
- b. Options specified during `vlog` — for example, `vlog ... +initreg+0`
- c. Global options specified during `vopt` — `vopt ... +initreg+Z`

This argument does not override any variable declaration assignment, such as:

```
reg r = 1'b0
```

- **+initwire[+{0 | 1 | X | Z}]**

(optional) Initializes unconnected nets in the interface, module, program, or package to the specified value, whether the nets are declared as ports or normal variables. Supports wire, wor, wand, trior, and triand net types. The default is 0 when +initwire is entered with no value. If +initwire is not used, unconnected nets default to Z.

- **-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 using 0). Setting the argument to a positive integer greater than one ($n > 1$) can improve vopt performance. The default is for vopt to use 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.

- **-L <libname>**

(optional) Searches the specified resource library for precompiled modules. The library search 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](#)" in the User's Manual for more information.)

Note



Questa SIM issues a warning if there are any inconsistencies between the library search you specify for vopt -L or vopt -Lf and the search you specified for the -L or -Lf arguments of the [vlog](#) command.

- **-Ldir <pathname> [<pathname> ...]**

(optional) Passes one or more container folders for libraries specified by either vsim -L or vsim -Lf. Once you specify a container folder (pathname), the libraries contained in this folder can be directly referenced using their logical names. Questa SIM searches multiple paths in the order in which you specify them on the command line.

- **-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 in the User's Manual and [Liberty Library Models](#) in the Power Aware Simulation User's Manual for more information.

<file_name> — A comma-separated list of liberty file names.

- **-libverbose[=prlib]**

(optional) Creates verbose messaging about library search and resolution operations.

prlib — prints the -L or -Lf option that was used to locate each design unit loaded by vopt or vsim. This information is printed to the right of the existing "– Loading design unit xyz..." messages.

Libraries containing top design units that are not explicitly present in the set of -L/-Lf options are implicitly promoted to searchable libraries at the end of the library search order. They appear as -Ltop in the output of the -libverbose option. To stop creation of -Ltop libraries, use option [-noltop](#) of vopt or vsim.

- **-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 are initialized before any subclass properties.

- **-logfile <filename>**

(optional) Generates a log file of the optimization.

-logfile <filename> — Saves transcript data to <filename>. Can be abbreviated to -l <filename>. Overrides the default transcript file creation set with the TranscriptFile and BatchTranscript *modelsim.ini* variables. (Refer to [TranscriptFile](#) or [BatchTranscriptFile](#) in the User's Manual.) You can also specify "stdout" or "stderr" for <filename>.

- **+maxdelays**

(optional) Selects maximum delays from the "min:typ:max" expressions. If preferred, you can use fsim -sdfmax to defer delay selection until simulation time.

If you specify the +mindelays, +typdelays, or +maxdelays flag with vopt, and specify a different flag with vsim, the simulation can use the delay value based upon the flag specified with vopt.

- **-memopt[=n]**

Enables or disables peak memory optimization.

Mode "n" values are 0, 1, 2, 3, 4.

Mode 0 — Disables this optimization.

Mode 1 & 2 — Enables less aggressive optimizations than mode 3 (the default mode) and may result in slightly higher peak memory.

Mode 3 — (default) Enables aggressive optimizations that may result in better peak memory.

- **+mindelays**

(optional) Selects minimum delays from the "min:typ:max" expressions. If preferred, you can use vsim -sdfmin to defer delay selection until simulation time.

If you specify the +mindelays, +typdelays, or +maxdelays flag with vopt, and specify a different flag with vsim, the simulation can use the delay value based upon the flag specified with vopt.

- **-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
```

- **-msglimit {error | warning | all[,-<msgNumber>,...] | none[,+<msgNumber>,...] | <msgNumber>,[msgNumber,...]}**

(optional) Limits messages to the default message limit count of five. Use the -msglimitcount argument to change the default message limit count. Specify multiple messages as a comma-separated list.

error — Stops the run when the total number of errors reaches the default count.

warning — Stops the run when the total number of warnings reaches the default count.

all — Limits all messages to the default count except those you specifically exclude. To exclude messages from the limit, supply a comma-separated list of message numbers, each preceded by a minus sign (-), for example “all,-<msgNumber1>,-<msgNumber2>,...”.

none — Excludes all messages from the default count except those you specifically include. To include messages to limit to the default count, supply a comma-separated list of message numbers, each preceded by a plus sign (+), for example “none,+<msgNumber1>,+<msgNumber2>,...”.

<msgNumber>[,<msgNumber>,...] — Specifies messages to limit to the default count.

Examples:

- Stop the run when the total number of errors reaches the default count:

```
-msglimit error
```

- Stop the run when the total number of warnings reaches the default count:

```
-msglimit warning
```

- Specifically limit messages 2374 and 2385 to the default count:

```
-msglimit 2374,2385
```

- Limit all messages to the default count, except messages 2374 and 2385:

```
-msglimit all,-2374,-2385
```

- Limit only messages 2374 and 2385 to the default count:

```
-msglimit none,+2374,+2385
```

- **-msglimitcount <limit_value>** **-msglimit [all|none,] [-|+]<msgNumber>[,-|+]<msgNumber>...]**

(optional) Limits the reporting of listed messages to the user-defined limit_value. Overrides the MsgLimitCount variable in the *modelsim.ini*. Refer to **MsgLimitCount** in the User's Manual.

- **-msglimitcount <limit_value>** **-msglimit [error|warning]**

(optional) Stops the run when the total number of reported errors/warnings reaches the user-defined limit_value.

- **-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 is a forward slash (/).

- **+nocheck<enum>**

(optional) Enables cell-level optimization on cells with various types of design characteristics, where <enum> determines the type to be checked, as listed below.

Note

 Optimization on all cells is enabled by default—this argument restores optimization on cells that you have used the **+check<enum>** argument on. You can specify any one or all of these behaviors to apply optimization on those cells.

ALL — Applies all +nocheck modes described below.

AWA — Allows an Always block to drive another Always block or a sequential UDP.

CLUP — Allows optimization of cells containing connectivity loops.

DELAY — When used in conjunction with vopt +delay_mode_path (see below), allows optimization of Verilog modules with distributed delays and no path delays in favor of module inlining.

DNET — Allows both the port and the delayed port (created for negative setup/hold) to be used in the functional section of the cell.

INBUF — Allows optimization of cells with input buffers.

INTRI — Allows inputs of type tri1 or tri0 in cell optimizations.

IPDOP — Allows procedural code where an input directly drives an output.

OPRD — Allows an output port to be read internally by the cell.

SUDP — Allows a sequential UDP to drive another sequential UDP.

- **-noconstimedassert**

(optional) Turns off the display of immediate assertions with constant expressions in the GUI, in reports, and in the UCDB. By default, displays immediate assertions with constant expressions. You can also set the ShowConstantImmediateAsserts variable in the vopt section of the *modelsim.ini* file to 0 (off).

- **-nocoverfecsingleterm**

Disable coverage for expressions that contain multi-bit sub-expressions but whose resulting output is still single bit.

- **-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. Refer to [CreateLib](#) in the User's Manual.

- **-nodebug[=ports | =pli | =ports+pli]**

(optional) Hides, in 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 is required for instantiation in a parent scope.

The design units' source code, internal structure, registers, nets, and so on, do not display in the GUI. In addition, you cannot access the hidden objects 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 — Combines the behavior of =ports and =pli.

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 ‘celldesignate’ compiler directive, thus marking them as cells (refer to the -v and -y arguments of vlog, which enable source library search). Use of 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 ‘celldesignate’ compiler directive. This argument restores the default library search behavior after it has been changed with the **+libcell | +nolibcell** argument.
- **-nologo**
(optional) Disables the startup banner.
- **-noltop**
Stops creation of -Ltop libraries. Libraries containing top design units that are not explicitly present in the set of -L/-Lf options are implicitly promoted to searchable libraries and placed at the end of the library search order. They appear as -Ltop in the output of the -libverbose argument unless you use the -noltop argument. See the **-libverbose[=plib]** option.
- **+nopathpulse**
(optional) Causes PATHPULSE\$ specparam(s) to be ignored in a module’s specify block. The default is to ignore PATHPULSE\$ specparam(s).
- **-noscelab**
(optional) Disables vopt elaboration of SystemC design hierarchy and HDL sub-hierarchy, if any exists. Use this argument 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[+<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 simulates the memory normally. Refer to [Sparse Memory Modeling](#) in the User's Manual for more information.

`+<selection>[+<recurse_level>|.]` — enables access for specific Verilog design objects and/or regions.

`<selection>` — Can take one of the following forms:

- `[<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.
- `<instance_path>` — A path to an instance.
- `<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 for selection to 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 that this argument causes `$sdf_annotation()` to be ignored, producing a suppressible error message.
- **-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 [note](#) and “[Message Severity Level](#)” in the User’s Manual.
- **+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>. You can disable warnings that include the <CODE> name in square brackets in the warning message. For example,

```
** Warning: test.v(15): [RDGN] - Redundant digits in numeric literal.
```

Specifying +nowarnRDGN disables this message.
- **-nowarn <category_number>**
(optional) Selectively disables a category of warning message. Allows multiple -nowarn switches. You can disable warnings for all compiles with the Main window **Compile** > **Compile Options** menu command or the *modelsim.ini* file (refer to [modelsim.ini Variables](#) in the User’s Manual).

The following table describes the warning message categories:

Table 3-12. 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 cannot 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, to increase your debugging visibility on a specific cell, and to place breakpoints on source lines that have been optimized out.
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 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](#)” on page 35” for more information.
- -override_precision
(optional) Used with the -timescale argument, this argument overrides the precision of `timescale specified in the source code.
- -pa_checks=<check_options>
(optional) Enables static RTL and dynamic checks. If you do not specify any check options, all dynamic checks are enabled.
 - -pa_checks=all — Enables all static RTL checks (except static RTL path analysis checks) and dynamic checks.
 - -pa_checks=s — Enables all static RTL checks, except static RTL path analysis checks. You have to explicitly enable static RTL path analysis checks.
 - -pa_checks=d — Enables all dynamic checks.

To selectively enable static RTL and dynamic checks use the following options:

Note



To specify more than one <check_options>, use the + operator between the options. For example:

```
vopt -pa_checks=rop+cp+smi
```

smi — (Static RTL isolation check) Reports missing isolation cell when an isolation cell is required for the power domain crossing, but an isolation cell is not present in the design, and the tool did not infer one.

Mnemonic: ISO_MISSING

sri — (Static RTL isolation check) Reports not required isolation cell when an isolation cell is not required for the power domain crossing, but either an isolation cell is present in the design, or the tool infers one.

Mnemonic: ISO_NOT_REQUIRED

ssi — (Static RTL isolation check) Reports incorrect isolation cell when an isolation cell is required for the power domain crossing, but you specify the set_isolation -no_isolation command in the UPF file.

Mnemonic: ISO_INCORRECT or ISO_INCORRECT_SUPPLY

svi — (Static RTL isolation check) Reports valid isolation cell when an isolation cell is required for the power domain crossing, and either an isolation cell is present in the design, or the tool infers one.

Mnemonic: ISO_VALID

sni — (Static RTL isolation check) Reports not analyzed isolation cell when the power state table (PST) information is not sufficient to analyze whether an isolation cell is required or not for the power domain crossing.

Mnemonic: ISO_NOT_ANALYZED

sdi — (Static RTL isolation check) Reports not inserted isolation cell when you specify the isolation strategy with the set_isolation -no_isolation command in the UPF file. However, an isolation cell may or may not be required for the power domain crossing.

Mnemonic: ISO_NOT_INSERTED

si — (Static RTL isolation check) Reports all static RTL isolation checks (smi, sri, sii, svi, sni, and sdi).

sml — (Static RTL level shifter check) Reports missing level shifter cell when a level shifter cell is required for the power domain crossing, but a level shifter cell is not present in the design, and the tool did not infer one.

Mnemonic: LS_MISSING

srl — (Static RTL level shifter check) Reports not required level shifter cell when a level shifter cell is not required for the power domain crossing, but either a level shifter cell is present in the design, or the tool infers one.

Mnemonic: LS_REDUNDANT

sil — (Static RTL level shifter check) Reports incorrect level shifter cell when the direction of the level shifter cell specified in the UPF file does not match the direction as determined by the voltage difference of the power domain crossing. For example, the tool reports an incorrect level shifter cell if the power domain crossing requires a low_to_high level shifter cell, and you specify the level shifter strategy in the UPF file as high_to_low.

Mnemonic: LS_INCORRECT

svl — (Static RTL level shifter check) Reports valid level shifter cell when a level shifter cell is required for the power domain crossing, and either a level shifter cell is present in the design, or the tool infers one.

Mnemonic: LS_VALID

snl — (Static RTL level shifter check) Reports not analyzed level shifter cell when the power state table (PST) information is not sufficient to analyze whether a level shifter cell is required or not for the power domain crossing.

Mnemonic: LS_NOT_ANALYZED

sdl — (Static RTL level shifter check) Reports not inserted level shifter cell when you specify the level shifter strategy with the set_level_shifter -no_shift command in the UPF file. However, a level shifter cell may or may not be required for the power domain crossing.

Mnemonic: LS_NOT_INSERTED

sl — (Static RTL level shifter check) Performs all static RTL level shifter checks (sml, srl, sil, svl, snl, and sdl).

snpl — (Static RTL path analysis check) Reports not analyzed path for level shifter requirements when the power domain crossing is not analyzed for the level shifter requirements because of insufficient PST information on either the source or sink supplies.

Mnemonic: LS_PATH_NOT_ANALYZED

scpl — (Static RTL path analysis check) Reports good path with no level shifter requirement when the power domain crossing is analyzed for the level shifter requirements, and no level shifter cell is required for the crossing.

Mnemonic: LS_PATH_GOOD

snpi — (Static RTL path analysis check) Reports not analyzed path for isolation requirement when the power domain crossing is not analyzed for the isolation requirements because of insufficient PST information on either the source or sink supplies.

Mnemonic: ISO_PATH_NOT_ANALYZED

scpi — (Static RTL path analysis check) Reports good path with no isolation requirement when the power domain crossing is analyzed for the isolation requirements, and no isolation cell is required for the crossing.

Mnemonic: ISO_PATH_GOOD

snpl+snpi — (Static RTL path analysis check) Reports not analyzed path for isolation and level shifter requirements when the power domain crossing is not analyzed for the isolation and level shifter requirements because of insufficient PST information on either the source or sink supplies.

Mnemonic: PATH_NOT_ANALYZED

scpl+scpi — (Static RTL path analysis check) Reports good path with no isolation and level shifter requirements when the power domain crossing is analyzed for the isolation and level shifter requirements, and no isolation and level shifter cell are required for the crossing.

Mnemonic: PATH_GOOD

s+b2b — (Static RTL back-to-back check) Reports the following back-to-back cells present in your design:

- Back-to-back isolation cells with same control signals for a power domain crossing.

Mnemonic: ISO_B2B_CTRL_SAME

- Back-to-back isolation cells with different control signals for a power domain crossing.

Mnemonic: ISO_B2B_CTRL_DIFF

- Back-to-back isolation cells with unknown control signals for a power domain crossing.

Mnemonic: ISO_B2B_CTRL_UNK

icp — (Dynamic isolation check) Reports violation if the clamp value of the isolation cell is different than the clamp value specified in the UPF file during the active isolation period.

Mnemonic: QPA_ISO_CLAMP_CHK

idp — (Dynamic isolation check) Reports violation if the isolation control signal is disabled when the source domain is OFF and sink domain is ON.

Mnemonic: QPA_ISO_DIS_PG

iep — (Dynamic isolation check) Reports violation if the isolation control signal is not enabled when the source domain is OFF and sink domain is ON.

Mnemonic: QPA_ISO_EN_PSO

idpcoa — (Dynamic isolation check) Reports violation if the isolation control signal is disabled when the source domain is in any of the following CORRUPT simstates, and the sink domain is not in a CORRUPT simstate.

- CORRUPT
- CORRUPT_ON_ACTIVITY
- CORRUPT_STATE_ON_CHANGE
- CORRUPT_STATE_ON_ACTIVITY

Mnemonic: QPA_ISO_DIS_PG

iepcoa — (Dynamic isolation check) Reports violation if the isolation control signal is not enabled when a source domain is in any of the following CORRUPT simstates, and the sink domain is not in a CORRUPT simstate.

- CORRUPT
- CORRUPT_ON_ACTIVITY
- CORRUPT_STATE_ON_CHANGE
- CORRUPT_STATE_ON_ACTIVITY

Mnemonic: QPA_ISO_EN_PSO

ifc — (Dynamic isolation check) Reports violation if the value at the output of an isolation cell is different from that at its input during the inactive isolation period.

Mnemonic: QPA_ISO_FUNC_CHK

ira — (Dynamic isolation check) Reports violation if there is no isolation requirement (such as the source domain is not in an OFF or CORRUPT state when the sink domain is in an ON state), and the isolation control signal is active.

Mnemonic: QPA_ISO_REDUNDANT_ACT

irc — (Dynamic isolation check) Reports violation if the value on the isolated port changes when its isolation control signal is changing state (high_to_low or low_to_high).

Mnemonic: QPA_ISO_PORT_TOGGLE

it — (Dynamic isolation check) Reports violation if the value on the isolated port changes during the active isolation period.

Mnemonic: QPA_ISO_ON_ACT

umi — (Dynamic isolation check) Reports violation if the isolation strategy is not specified for a power domain crossing whose source domain is ON and sink domain is OFF.

Mnemonic: QPA_UPF_MISSING_ISO_CHK

i — (Dynamic isolation check) Reports the following dynamic isolation checks: icp, idp, iep, idpcoa, iepcoa, ifc, ira, and irc.

uml — (Dynamic level shifter check) Reports violation if a level shifter strategy is not specified in the UPF file when a level shifter cell is required for the power domain crossing. Also, strategy specified with set_level_shift -no_shift in the UPF file are checked for any missing level shifter cells.

Mnemonic: QPA_UPF_MISSING_LS_CHK

uil — (Dynamic level shifter check) Reports violation if the direction of level shifter cell specified in the UPF file does not match the direction determined by the voltage difference of the power domain crossing. For example, the tool reports an incorrect level shifter cell if the power domain crossing requires a low_to_high level shifter cell and you specify the level shifter strategy in the UPF file as high_to_low.

Mnemonic: QPA_UPF_INCORRECT_LS_CHK

ul — (Dynamic level shifter check) Reports all dynamic level shifter checks.

isa — (Dynamic level shifter check) 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 the isolation enable signal. (Not supported for UPF)

Mnemonic: ISO_MISSING

rop — (Dynamic retention check) Reports violation if the retention condition is not asserted when the power is switched off. For a balloon-latch configuration, the check is not activated if the asynchronous set or reset is active.

Mnemonic: QPA_RET_OFF_PSO

rpo — (Dynamic retention check) Reports violation if there is an error in the sequence of triggering of the retention condition and power signal. For retention to succeed, the retention condition must be high at power down and power up, and this check is triggered when the retention condition is not met. For a balloon-latch configuration, the check is not activated if the asynchronous set or reset is active.

Mnemonic: QPA_RET_PD_OFF

rcs — (Dynamic retention check) Reports violation if the clock or latch enable is not at a certain value when the save and restore events take place. If a latch is enabled and can change its value, triggering retention can potentially cause race conditions in the stored value. This is also a check against such conditions. This check is for balloon-latch configuration only; it does not apply to master-slave (slave-alive) retention.

Mnemonic: QPA_RET_CLK_STATE

rsa — (Dynamic retention check) Reports violation if the clock toggles during the retention period.

Mnemonic: QPA_RET_SEQ_ACT

rtcon — (Dynamic retention check) Reports violation if the retention condition is off when the power is enabled.

Mnemonic: QPA_RET_ON_RTC

rtcoff — (Dynamic retention check) Reports violation if the retention condition is off when the power is disabled.

Mnemonic: QPA_RET_OFF_RTC

rtctog — (Dynamic retention check) Reports violation if the retention condition toggles during power down. Toggling of the retention condition during power down corrupts the retained value.

Mnemonic: QPA_RET_RTC_TOGGLE

rtc — (Dynamic retention check) Reports the following retention condition checks: rtcon, rtcoff, and rtctog.

r — (Dynamic retention check) Reports all dynamic retention checks.

t — (Dynamic miscellaneous check) Reports violation if the input to a power domain toggles when the power domain is turned off.

Mnemonic: QPA_PD_OFF_ACT, and QPA_PD_BIAS_ACT

cp — (Dynamic miscellaneous check) Reports violation when the power signal to any power domain gets corrupted. This check does not flag a violation when both the source and sink power domain supplies are off. For UPF, this check is applicable to control ports of a switch, isolation enable signal of an isolation strategy, and retention save and restore signals of a retention strategy.

Mnemonic: QPA_CTRL_SIG_CRPT

p — (Dynamic miscellaneous check) Reports the power domains that are switched on or off.

Mnemonic: QPA_PD_STATUS_INFO

pis — (Dynamic miscellaneous check) Reports violation if there is an undefined or illegal state on a PST or supply port. A state on a PST is undefined or illegal if the state of nets or ports is not a valid combination as related to the power state table in the UPF file. A state on a supply port is undefined or illegal if it is not defined with add_port_state for the supply port.

Mnemonic: QPA_UPF_ILLEGAL_STATE_REACHED

ugc — (Dynamic miscellaneous check) Reports any spurious spikes (glitches) on control lines so that they do not cause false switching of the control ports of control logic (such as isolation, power switch, and retention). Use the pa msg - glitch_window command to specify the maximum allowed time window of the glitch.

Mnemonic: QPA_CTRL_SIG_GLITCH

npu — (Dynamic miscellaneous check) By default, reports violation if the non-retention registers are not reset when the power domain containing them is powered up. It also generates the report file *report.nretsyncff.txt*.

When you define the attribute qpa_attr_inactive_reset_duration and/or qpa_attr_active_reset_duration in the set_design_attributes command, then the simulator reports the following:

- Reports violation if the asynchronous reset is not asserted within Tmax time units after power-up. See “qpa_attr_inactive_reset_duration” in the “[Supported UPF Attributes](#)” section of the Power Aware User’s Manual.
- Reports violation if the asynchronous reset is not asserted for minimum Twidth time units. See “qpa_attr_active_reset_duration” in the “[Supported UPF Attributes](#)” section of the Power Aware User’s Manual.

Mnemonic: QPA_NRET_ASYNCFF

upc — (Dynamic miscellaneous check) Reports violation if the isolation or retention supplies are switched off during the active isolation or retention period.

Mnemonic: QPA_UPF_PG_CHK

- **-pa_checkoption=<value>**

(optional) Global settings that affect all static RTL and dynamic checks that are enabled with the -pa_checks argument. Where <value> can be:

assertionhierpath — Displays the full hierarchical path of the failing checks in the dynamic report, when the dynamic checks are enabled.

- **-pa_connectpgpin=i | a | e | c**

(optional) Ignores the connect_supply_net UPF command to connect supply net to port when the port is missing in the verification model, but 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 connect supply net to these ports, and disables the Power Aware simulation semantics of the parent instance of the port.
- e — Flags error message for these ports (default).
- c — Enables the creation of a port when connect_supply_net identifies a port that is missing in the HDL source, 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_corrupt=integer — (default) Enables corruption of integer types.
 - pa_nocorrupt=real — (default) Disables corruption of real types.
 - pa_nocorrupt=integer — Disables corruption of integer types.
 - -pa_coverage [=checks] [=crosscov] [=crossdontcare] [=implicitportnet] [=iso] [=portpststate] [=powerstate] [=ret] [=simMode] [=switch] [=undeftrans]
(optional) Enables coverage for Power Aware simulation. Use the following options with the -pa_coverage argument to selectively enable Power Aware coverage:

Note

 Specify the -pa_coverage argument without any option to enable all options, except implicitportnet and undeftrans.

checks — Enables coverage of Power Aware dynamic checks.

crosscov — Enables cross coverage of combination of power domains that are interconnected through power states (you can also use the vopt -pa_enable=crosscoverage option to enable cross coverage). The simulator enables cross coverage when you enable coverage of power states using the -pa_enable=powerstate option.

crossdontcare — Enables coverage of don't care states of power domains during cross coverage. Don't care states are the states whose dependency you do not explicitly specify in the add_power_state command.

Note

 To enable coverage of don't care states, you have to enable default cross coverage with the -pa_coverage=powerstate option.

implicitportnet — Enables coverage of implicitly created state (by the simulator) for supply ports, supply nets, and/or supply set handle functions that are used in the -supply_expr argument of the add_power_state command.

iso — Enables coverage of the following:

- Predefined states and transitions of the simstates of the isolation supply set.

- Predefined states and transitions of the isolation enable signal:
 - Predefined states — ACTIVE_LEVEL, INACTIVE, ACTIVE_X and ACTIVE_Z
 - Predefined transitions — HIGH_TO_LOW and LOW_TO_HIGH
- portpststate — Enables coverage of user-defined states and transitions of ports, nets, and power state tables.
- powerstate — Enables coverage of the following:
- User-defined and predefined states and transitions of supply sets
 - User-defined states and transitions of power domains
 - Cross coverage of states of power domains
- ret — Enables coverage of the following:
- Predefined states and transitions of the simstates of the retention supply set
 - Predefined states and transitions of the save and restore events:
 - Predefined states — ACTIVE_LEVEL, and INACTIVE
 - Predefined transitions — HIGH_TO_LOW, and LOW_TO_HIGH
- simMode — Enables coverage of states and transitions of simstates of the following:
- Primary supply set of the power domain
 - Isolation supply set of the isolation strategy
 - Retention supply set of the retention strategy
- switch — Enables coverage of the following:
- Predefined states and transitions of the switch
 - Predefined states and transitions of the control port and acknowledge (ack) port of the switch:
 - Predefined states — ACTIVE_LEVEL, INACTIVE, ACTIVE_X and ACTIVE_Z
 - Predefined transitions — HIGH_TO_LOW, and LOW_TO_HIGH
- undeftrans — Enables coverage of unnamed and undefined transitions of a port, net, power domain, supply set, power switch, or PST state when the add_state_transition or describe_state_transition command is specified in the UPF file.
- -pa_coverageoff [=checks] [=crosscov] [=crossdontcare] [=implicitportnet] [=iso] [=portpststate] [=powerstate] [=ret] [=simMode] [=switch] [=undeftrans]
 - (optional) Disables coverage for Power Aware simulation. This is the default mode.

Use the following options with the `-pa_coverageoff` argument to selectively disable Power Aware coverage:

Note

 Specify the `-pa_coverageoff` argument without any option to disable all options.

`checks` — Disables coverage of Power Aware dynamic checks.

`crosscov` — Disables cross coverage of combination of power domains that are interconnected through power states.

`crossdontcare` — Disables coverage of don't care states of power domains during cross coverage. Don't care states are the states whose dependency you do not explicitly specify in the `add_power_state` command.

`implicitportnet` — Disables coverage of implicitly created state (by the simulator) for supply ports, supply nets, and/or supply set handle functions that are used in the `-supply_expr` argument of the `add_power_state` command.

`iso` — Disables coverage of the following:

- Predefined states and transitions of the simstates of the isolation supply set.
- Predefined states and transitions of the isolation enable signal:
 - Predefined states — ACTIVE_LEVEL, INACTIVE, ACTIVE_X and ACTIVE_Z
 - Predefined transitions — HIGH_TO_LOW and LOW_TO_HIGH

`portpststate` — Disables coverage of user-defined states and transitions of ports, nets, and power state tables.

`powerstate` — Disables coverage of the following:

- User-defined and predefined states and transitions of supply sets
- User-defined states and transitions of power domains
- Cross coverage of states of power domains

`ret` — Disables coverage of the following:

- Predefined states and transitions of the simstates of the retention supply set
- Predefined states and transitions of the save and restore events:
 - Predefined states — ACTIVE_LEVEL, and INACTIVE
 - Predefined transitions — HIGH_TO_LOW, and LOW_TO_HIGH

`simMode` — Disables coverage of states and transitions of simstates of the following:

- Primary supply set of the power domain
- Isolation supply set of the isolation strategy

-
- Retention supply set of the retention strategy

switch — Disables coverage of the following:

- Predefined states and transitions of the switch
- Predefined states and transitions of the control port and acknowledge (ack) port of the switch:
 - Predefined states — ACTIVE_LEVEL, INACTIVE, ACTIVE_X and ACTIVE_Z
 - Predefined transitions — HIGH_TO_LOW, and LOW_TO_HIGH

undeftrans — Disables coverage of unnamed and undefined transitions of a port, net, power domain, supply set, power switch, or PST state when the add_state_transition or describe_state_transition command is specified in the UPF file.

- **-pa_crosscoveredepth <integer>**

(optional) Specifies the number of levels of power domains that receive cross coverage. The default value of *<integer>* is 1.

- **-pa_db <path_file_name>**

(optional) Specifies a location and filename for the Power Aware database, which you use with Mentor Visualizer Debug Environment. For example:

```
vopt -pa_db ./database/testing.bin.padb
```

The default location is the current working directory, and the default filename is *design.bin.padb*.

- **-pa_defertop**

(optional) Enables you to change the root of DUT hierarchy to a new pathname during simulation, without rerunning vopt.

Note

 You do not need to use the vsim -pa_top=<pathname> command with this argument, though using them together is supported for backward compatibility.

-
- **-pa_dbgfindobj=<filename>**

(optional) Writes the results of all find_objects UPF commands to the specified *<filename>*.

- **-pa_disable=<value>[+<value>]...**

(optional) Disables one or more actions that Questa SIM performs during Power Aware simulation. Use these values to control compatibility with previous releases.

<value> — One or more values, as listed in “[Usage of vopt -pa_enable and -pa_disable](#)” in the *Power Aware Simulation User’s Manual*.

Note

 To specify more than one value for the `-pa_disable` argument, use the plus (+) operator between the values. For example:

```
vopt -pa_disable=insertls+insertiso+sourcesink
```

- `-pa_dumpimdb`
(optional) Generates the Power Aware database file, *design.bin.padb* at the current working directory, or to the library location you specify with the `-pa_lib` argument to `vopt`. You can then load this database for simulation with the `vsim -pa_loadimdb` command, which makes the Power Aware information model available for UPF query commands in the UPF file.
- `-pa_dumplibertydb=<database_filename>`
(optional) Specifies 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 the specified 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 one or more actions that Questa SIM performs during Power Aware simulation. Use these values to control compatibility with previous releases.

<value> — One or more values, as listed in “[Usage of vopt -pa_enable and -pa_disable](#)” in the *Power Aware Simulation User’s Manual*.

Note

 To specify more than one value for the `-pa_enable` argument, use the plus (+) operator between the values. For example:

```
vopt -pa_enable=insertls+insertiso+sourcesink
```

- `-pa_excludefile <filename>`
Skips Power Aware processing of a module, instances, or signals of a module within a particular hierarchical path.

<filename> — (required) The name of a text file that contains modules, instances, or signals of a module that you want to exclude from Power Aware processing. Each entry in the file must be of the following form:

`<module_name> [-a] [<hier_path>] [-s | -sr <signal_name>]`

<module_name> — (required) Excludes the module, *<module_name>*, from Power Aware processing. To specify any regular expression in *<module_name>*, use quotation marks.

Note

 Power Aware processing does not exclude any module instantiated within *<module_name>*.

-a — (optional) Enables recursive exclusion of all instances within *<module_name>*. If you specify this argument, Questa SIM skips Power Aware processing of all instances present within *<module_name>*.

<hier_path> — (optional) Excludes the instance of *<module_name>* present at *<hier_path>* from Power Aware processing. To specify any regular expression in *<hier_path>* use quotation marks.

When Power Aware processing excludes an instance of *<module_name>*, Questa SIM displays the following message:

```
** Note: (vopt-9691) Excluding power aware module '<module_name>' in path '<hier_path>'.
```

When you use *<hier_path>* with the -a argument, *<hier_path>* limits the recursive exclusion to a particular scope.

-s *<signal_name>* — (optional) Excludes *<signal_name>* present in *<module_name>* from Power Aware processing. To specify any regular expression in *<signal_name>*, use quotation marks.

-sr *<signal_name>* — (optional) Excludes *<signal_name>* present in *<module_name>* and in the generate block of *<module_name>* from Power Aware processing. To specify any regular expression in *<signal_name>*, use quotation marks.

Use the pound sign (#) to begin any comment lines in the file.

- -pa_genrpt= [ud] [pa[+b]] [de[+b]] [cell] [conn] [srcsink]

(optional) Generates the Power Aware reports. Specifying the -pa_genrpt argument without any value generates all reports.

Note

 To specify more than one report, use the + operator between the values. For example:

```
vopt -pa_genrpt=pa+cell+conn
```

ud — Generates the dynamic UPF report in the transcript window. The report contains the power domain status, supply net and supply port toggle information, time and polarity of controls (such as control port of a power switch, retention save and restore signals, isolation enable signal), and so on.

pa — Generates a report, *report.pa.txt*, about the Power Aware architecture (power domains, power switches, retention strategy, isolation strategy, level shifter strategy, PSTs) in your design.

de — Generates a report, *report.de.txt*, about power domain elements in your design, and the corresponding Power Aware information.

Note

 Specify the value “b” with the “pa”, and “de” values in the -pa_genrpt argument to include bitwise expanded information in the report. For example:

```
vopt -pa_genrpt=pa+b
```

cell — Generates a report, *report.cell.txt*, about the macro cells in your design.

conn — Generates a report, *report.connection.txt*, about the connections between your UPF, HDL and Liberty files.

srcsink — Generates a report, *report.srccsink.txt*, on source-sink-paths, analyzed for static checks analysis. Each reported path has a unique path-id-tag. The static checks report contains a corresponding path-id-tag for each reported check.

Refer to “[Power Aware Reports Reference](#)” in the *Power Aware Simulation User’s Manual* for more information.

- -pa_gls

(optional) Enables gate-level simulation for a Power Aware analysis. Use the -pa_gls argument only when you run a gate-level simulation, and not when you run a mixed RTL and gate-level simulation.

- -pa_glschecks=<check_options>

(optional) Enables static GLS checks.

<check_options> has any of the following values:

s — Reports missing or inconsistent isolation, level shifter, retention, and switch cells present in the design.

s+els — Reports back-to-back ISO-LS, and LS-ISO cells present in the design.

s+b2b — Reports back-to-back isolation, and level shifter cells present in the design.

Refer to “[Static GLS Checks](#)” in the *Power Aware Simulation User’s Manual* for more information.

- -pa_hiersep <alphanum_character>

(optional) Specifies that the UPF file uses a hierarchical path separator different than the default slash (/) character.

- -pa_incrptval0

(optional) Sets the default corruption value to 0 (zero) instead of ‘left’.

- -pa_interactive

(optional) Enables running Power Aware simulation multiple times, without reloading the design for each successive run. Using this argument results in a “PA>” prompt appearing on the command line, which you can use as any Tcl or shell prompt. You can now run the Questa SIM commands (such as another vopt command or pa report) without reloading the design.

To exit this interactive mode, enter quit or exit.

- **-pa_lib <library_pathname>**
 (optional) Specifies a library location in which to store the Power Aware database file, *design.bin.padb*. By default, the database file is stored in the current working directory. Use a different library location if you want to preserve the Power Aware verification results of the same design for different simulation modes. Invoke the vsim command with the **-pa_lib** argument to specify the library location to use.
- **-pa_libertyfiles=<liberty_filename>[,<liberty_filename>...]**
 (optional) Specifies a comma-separated list of Liberty files. The simulator parses the Liberty files in the order in which they are specified in the **-pa_libertyfiles** argument. If there are same libraries in different Liberty files, then the database contains the contents of the first parsed Liberty file. Refer to “[Liberty Library Models](#)” in the *Power Aware Simulation User’s Manual* for more information.
- **-pa_libertyNestedCmnts**
 (optional) Enables vopt to parse nested comments in Liberty files.
- **-pa_libertyrefresh=<database_filename>**
 (optional) Refreshes old library dumps in the cache directory created with vopt **-pa_loadlibertydb** or **-pa_dumplibertydb**.
- **-pa_libertyupdate**
 (optional) Updates the Liberty database (which contains dumps of the previous vopt run), with the dumps of the current vopt run. The current vopt run uses the liberty files specified with the **-pa_Libertyfiles** argument. If there are same libraries in different liberty files, then the database contains the contents of the last parsed liberty file.
- **-pa_loadlibertydb=<database_filename>**
 (optional) Loads a user-defined Liberty database created from a previous vopt run for the Power Aware analysis. The simulator uses this database 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 flags 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](#) (refer to the *Power Aware Simulation User’s Manual* for more information). Default value is 8.

- **-pa_noinpcorr**
(optional) 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 “[set_partial_on_translation](#)” in the *Power Aware Simulation User’s Manual*.
- **-pa_opt=nogls**
(optional) Disables Power Aware processing on design units that are not gate-level models.

Note

 This is a deprecated argument.

- **-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 to search for the supplies referenced in the UPF package functions supply_on or supply_off if the supplies are not found in the scope of calling function or in the scope of **-pa_top**.
- **-pa_prefix <hier_path>**
(optional) Specifies a hierarchical path to prepend to the path information in the UPF file. The path in a UPF file starts from a specific portion of the design. Use this argument to create a full path based on the top module on which you run the vsim command.

Note

 This is a deprecated argument. Use **-pa_top** instead.

- **-pa_preupffile <filename>**
(optional) Specifies a Tcl file that is processed before processing the UPF file. The Tcl file contains UPF and Tcl commands:
 - UPF Commands — All UPF commands supported by Questa SIM. Refer to “[Supported UPF Commands](#)” in the *Power Aware Simulation User’s Manual* for more information.
 - Tcl Commands — All Tcl commands supported by Questa SIM. Refer to “[Tcl Commands](#)” in the *Power Aware Simulation User’s Manual* for more information.

Tip

 Use the **-pa_tclfile <filename>** argument to specify a Tcl file that is processed after processing the UPF file.

- **-pa_pstcompflags=[p | g | pg | pstids | useallps]**
 (optional) Controls the behavior of PST composition, which is enabled with **-pa_enable=pstcomp**.

Note

 Use a plus (+) operator to combine the flags:

```
-pa_pstcompflags=p+pstids
```

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 — Adds 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 “[Power State Tables](#)” in the *Power Aware Simulation User’s Manual*.

- **-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 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.

Note

 This is a deprecated argument. Use **-pa_top** instead.

- **-pa_reportdir <pathname>**

(optional) Specifies an alternate directory in which to store the Power Aware reports. The default location is the **pa_reports** directory under the current directory:

```
<current_directory>/pa_reports
```

- **-pa_rslvnetcheck**

(optional) Enables a check 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 [set_partial_on_translation](#) in the *Power Aware Simulation User’s Manual*.

- **-pa_rtlinfo**
(optional) Runs Power Aware simulation in verbose mode. This helps identify register/latch modeling errors, such as non-synthesizable clocking styles.
- **-pa_staticchecksonly**
(optional) Performs only static checks on the design, without performing the optimization and code generation tasks, which results in a faster execution of the vopt command. Also generates static check reports that you can view with the pa report command. Once you have analyzed and verified the results, run vopt without this argument to proceed to a full vopt analysis and simulation.

Use the **-pa_staticchecksonly** argument with the **-pa_checks** argument to limit your static checks. For example, adding the following arguments to your standard vopt command enables isolation static checks only:

```
vopt <standard_options> -pa_staticchecksonly -pa_checks=si
```

- **-pa_tclfile <filename>**
(optional) Specifies a Tcl file that is processed after processing the UPF file. The Tcl file contains UPF and Tcl commands:
 - UPF Commands — All UPF commands supported by Questa SIM. Refer to “[Supported UPF Commands](#)” in the *Power Aware Simulation User’s Manual* for more information.
 - Tcl Commands — All Tcl commands supported by Questa SIM. Refer to “[Tcl Commands](#)” in the *Power Aware Simulation User’s Manual* for more information.

Tip

 Use the [**-pa_preupffile <filename>**](#) argument to specify a Tcl file that is processed before processing the UPF file.

- **-pa_top <pathname>**
(optional) Specifies the hierarchy of a DUT to be used for Power Aware analysis of a UPF file. This enables analysis of the UPF file from a hierarchy 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), you would specify the following:

```
-pa_top /tb/dut_inst
```
- **-pa_upf <filename>**
(optional) Specifies the UPF file name, which is used in the vopt run.
- **-pa_upfextensions=<extension>[+<extension>]**
(optional) Enables you to define and apply various UPF behaviors that the current UPF standard does not support.

For more information about the *<extension>* values, refer to “[Supported UPF Extensions](#)” in the *Power Aware Simulation User’s Manual*.

To specify more than one value, use the + operator between the values. For example:

```
vopt -pa_upfextensions=relatedsnet+genblk+v
```

Specifying this argument with no values is equivalent to specifying vopt -pa_upfextensions=default.

- **-pa_upflist <filename>**

(optional) Specifies a filename that lists multiple UPF files, which are used in the vopt run. For example:

```
vopt -pa_upflist ./UPF/upf.f
```

where *upf.f* contains the following files:

```
./UPF/connect_ports.upf
./UPF1/mem_config.upf
./UPF2/mem_const.upf
```

- **-pa_upfsanitychecks**

(optional) Checks for UPF syntax errors before loading the design, and reports errors encountered in applying low power intent on the design. In the case of no errors, Questa SIM continues to work as usual.

- **-pa_upfversion=[1.0 | 2.0 | 2.1 | 3.0]**

(optional) Specifies the version of UPF to use for Power Aware simulation.

1.0 — A subset of *IEEE Standard for Design and Verification of Low Power Integrated Circuits* (IEEE Std 1801-2009), originally developed by Accellera Organization, Inc; enables all semantics for UPF 1.0.

2.0 — (default) *IEEE Standard for Design and Verification of Low Power Integrated Circuits* (IEEE Std 1801-2009); enables all semantics for UPF 2.0.

2.1 — *IEEE Standard for Design and Verification of Low Power Integrated Circuits* (IEEE Std 1801-2013); enables all semantics for UPF 2.1.

3.0 — *IEEE Standard for Design and Verification of Low Power Integrated Circuits* (IEEE Std 1801-2015); enables all semantics for UPF 3.0.

- **-pa_zcorrupt**

(optional) Changes the default corruption value used at power-down from 'X' to 'Z'. This makes debugging easier. When you use the -pa_zcorrupt argument, Z represents the power-related corruption of a signal, and X represents initial values, logical conflicts that arise in the design due to errors, or X values intentionally specified in the design. At power-up, the design elements takes either the X value or their RTL value based on their logic types:

- Sequential elements takes the X value when the asynchronous controls are low, otherwise they take the RTL (reset) value.

- Combinational elements takes the RTL (reset) value.

This change affects the following types:

- VHDL — std_logic, std_uologic, and their aggregates.
- Verilog — all four state data types and their aggregates.

- **+pathpulse**

(optional) Enables usage of the PATHPULSE\$ specparam in a module's specify block. Has no effect on modules without PATHPULSE\$ specparam(s). The default is to ignore PATHPULSE\$ specparam(s).

- **-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 “[Preoptimizing Regions of Your Design](#)” in the User’s Manual for further information.

- **-pduignore[=<instpath>]**

Ignore Preoptimized Design Unit (black-box). If <instpath> is not specified, all PDUs found in compiled libraries are ignored. Otherwise, the PDU specified by <instpath> is ignored. You can specify this argument multiple times using different values of <instpath>.

- **-pdusavehierrefs[=<filename>]**

Creates a file with +acc settings. Required to maintain the necessary visibility of hierarchical references. By default the output filename is “mti_pdu_hier_refs”. You can 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 and places them in a file that vopt then uses 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](#) in the User’s Manual for more information.

Note

 Since the only intent of this step is to extract visibility requirements for the PDUs, vopt exits after writing the output files. Therefore, vopt does not require "-o" option when you specify -pduspec, 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 allows named port connections on bit-select and part-select ports, though only when the port list does not contain any multiple bit-select or part-select ports of the same name.

You can produce a complete list by executing the command:

```
verror -kind vopt -permissive
```

- **-[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>. Customer Support uses output from these arguments 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 read in during optimization. You can also specify a PSL file during compilation with the [vcom](#) command. Refer to “[Applying PSL Assertions During Elaboration/Optimization](#)” in the User’s Manual for more information.

- **-pslfile_vl <filename>**

(optional) Identifies a Verilog PSL file to read in during optimization. You can also specify a PSL file during compilation with the [vlog](#) command. Refer to “[Applying PSL Assertions During Elaboration/Optimization](#)” in the User’s Manual for more information.

- **-quiet**

(optional) Disables output of informative messages about processing the design, such as Loading, Compiling, or Optimizing, but not messages about the design itself.

- **-reporthrefs+ | -reporthrefs**

(optional) Extracts module, filename and line number information for HDL hierarchical references that cross into the specified module or instance boundary. Writes the information to an output file you specify with the "+f" option. You can then manually edit the flagged lines to disable such hierarchical access. Note that a hierarchical name (irrespective of where it is used in the hierarchy) will be extracted and reported as crossing into the specified module or instance if it contains any reference to the boundary module or instance. Any floating of parameters or generics is internally disabled in reporthrefs mode.

```
-reporthrefs+[[[<libName>.]<moduleName>] |  
[<instancePathName>]][+f=<fileName>] [+g]]
```

<libName>

Use this option to designate the library where the design has been compiled.

<moduleName>

Use this option to designate a module. You can designate a module name without designating a library.

<instancePathName>

Use this option to designate a fully rooted path to an instance.

+f=<fileName>

Use this option to designate an output filename.

+g

Use this option to force floating of parameters and generics.

-reporthrefs[[+i=<instancePathName>] [+m=[<libName>.]<moduleName>]]
[+f=<fileName>][+g][+a]

+i=<instancePathName>

Use the +i option to designate a fully rooted path to an instance.

+m=[<libName>.]<moduleName>

Use the +m option to designate a library and module name.

+f=<fileName>

Use the +f option to designate an output filename.

+g

Use the +g option to force floating of parameters and generics.

+a

Use the "+a" option to report all hierrefs with reference to specified boundary. By default, reports only hierrefs that cross the specified instance boundary.

The -reporthrefs argument replaces the -tbxhierrefs argument, and prints information for each hierarchical reference object type.

- **-rnmautointerconnect**

(optional) Enables conversion of wire objects used as structural wires in Verilog-AMS to interconnect nets defined by SystemVerilog. An interconnect object is a typeless net that you can use only to express connections; you cannot use it to express behavior. This option specifically converts wires connected upwards from real, wreal, nettype, and interconnect ports.

- **-source**

(optional) Displays the associated line of source code before each error message that is generated during compilation. By default, displays only the error message.

- **-sc22**
 (optional) Enables the SystemC-2.2 version (IEEE 1666-2005 standard) for compiling and linking. You can also enable SystemC-2.2 with the [Sc22Mode](#) *modelsim.ini* variable.
- **-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](#)” in the User’s Manual).
 If you specify multiple SystemC startup arguments, each must have a separate `-sc_arg` argument. SystemC startup arguments returned through `sc_argv()` are in the order in which they appear on the command line. You can include white space in a `<string>`; 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 necessary only when you compile the shared library 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) Limits the number of Verilog SDF missing instance messages to allow before terminating vsim. `<n>` is the maximum number of missing instance error messages to allow. The default value is 5.
- **-seq_udp_delay <value>[<units>]**
 Specifies a constant delay time for all sequential UDPs in the design, and sets all other structural delays to zero. The delay time is expressed as a value with an optional units specification. If you do not specify units, the argument uses the default unit.

- **-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 on 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 be the result of 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*.
- **-staticchecksmvdvhdl**
 (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).
 Specify multiple features and modes for each instance of -stats as a comma-separated list. You can specify -stats multiple times on the command line, but only the final instance takes 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 it without the plus (+) character. Setting this switch adds or subtracts features and modes from the settings in the Stats *modelsim.ini* variable.

<feature>

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.

Note



Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

You can set modes 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 kilobyte 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 {<msgNumber> | <msgGroup>} {[,<msg_number> | <msgGroup>] ,...]
(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 [suppress](#) and “[Message Severity Level](#)” in the User’s Manual for more information.

<msgNumber>

A specific message number

<msgGroup>

A value identifying a pre-defined group of messages, based on area of functionality. These groups suppress only notes or warnings. The valid arguments are:

All, GroupNote, GroupWarning, GroupFLI, GroupPLI, GroupSDF, GroupVCD, GroupVital, GroupWLF, GroupTCHK, GroupPA, GroupLRM

- -svext=[+|-]<extension>[,[+|-]<extension>]...

(optional) Enables SystemVerilog language extensions through a comma-separated list of arguments.

[+ | -] — controls activation of the *extension*. Arguments on the command line control the precedence; any settings to this switch override your settings of the SvExtensions *modelsim.ini* variable. Refer to [SvExtensions](#) in the User’s Manual.

+ — 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.

arif — Allow the use of refs in fork-join_any or fork-join_none blocks inside tasks.

atpi — Use type names as port identifiers. Disabled when compiling with -pedanticerrors.

basd — By default, always@(*) with delay or event usage inside the process, triggers on bit/part/field-select of the variable instead of the full variable. The basd option disables this default behavior and triggers on the full variable.

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 of 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
```

defervda — SV variables having an initializer in the declaration will trigger top-blocking always blocks at time zero.

dmsbw — Drives the MSB unconnected bits of the wider hiconn (output) or the wider loconn (input) in an otherwise collapsible port connection. With this extension, zero drives unconnected bits; otherwise, they float.

evis — Supports the expansion of environment variables within curly braces ({}) within `include string literals and in `include path names. For example, if MYPATH exists in the environment, it is 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.

ifslvbefr — Allows a solve/before constraint within an IfElse constraint with a constant condition. This is on by default.

omaa — Allows shuffle method on associative arrays.

pae — Automatically export all symbols imported and referenced in a package.

pae1 — Allows the export, using wildcard export only, of package symbols from a subsequent import of a package. These symbols may or may not be referenced in the exporting package.

[-]realrand — Enables randomization of ‘real’ variables and constraints, with the exception of multiply and divide with ‘real’ operands in constant expressions. This extension is enabled by default. Use “-svext=-realrand” to disable this extension at compile time.

Note

 Randomization of ‘real’ variables or constraints requires a SystemVerilog Real Number Modeling (svrnm) license. If the license check fails, ‘real’ number support is disabled.

sas — Enables LRM-compliant @* sensitivity rules.

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.

substr1 — Allows one argument in the builtin function substr. A second argument will be treated as the end of the string.

tzas — The always block will behave as though triggered at time zero even if none of the variables and nets in the implied sensitivity change value at time zero. This extension runs a top-blocking always @* at time zero, as is done for an always_comb.

ubdic — Allows the use of a variable in a SystemVerilog class before it is defined.
For example:

```
class A;
    function func();
        int y = x;           // variable 'x' is used before it is defined
        endfunction
        int x = 1;
    endclass
```

If you do not enable this extension, you will receive unresolved reference errors.

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.

Specify multiple extensions as a comma-separated list. For example:

vlog -svext=+fec,-uslt,pae

- -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; vopt uses this file to improve the visibility of PLI functions and vsim uses it to register the PLI functions.

-
- **-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 you must enclose <time_units / <time_precision> 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 \(Verilog\)](#)” in the User’s Manual 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; 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 because inlining is turned off. You can selectively enable toggle coverage on specific ports with the “vopt +acc=p+<selection>” command. Overrides any global value set by the TogglePortsOnly *modelsim.ini* variable. Refer to [TogglePortsOnly](#) in the User’s Manual.

- **-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. 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 can use the delay value based upon the flag you specify with vopt.

- **-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. Refer to [UdefSyms](#) in the User's Manual.

{<args>}

You must specify at least one argument.

on — (default) Enables automatic generation of stub definitions for undefined symbols and permits loading of the shared libraries despite the undefined symbols.

off — 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 [warning](#) and ["Message Severity Level"](#) in the User's Manual for more information.

- **-warning error**

(optional) Reports all warnings as errors.

- **-work <library_name>**

(optional) Specifies a logical name or pathname of a library to map to the logical library `work`. By default, the optimized output for the design is added to the work library. The pathname you specify overrides the pathname specified for work in the project file.

- **-xprop[,mode=resolve|pass|trap|none][,report=fatal|error|warning|info|suppress][,object=<hier_path> | <du>[/<inst>] | <hierlevel>]**

(optional) Defines the level of pessimism for X propagation for specified modules and/or instances. You can override the effect of this argument with the [xprop disable](#) command. This argument can take any of the following options:

mode=resolve (default) — evaluates all possible branches. 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.

mode=pass — if any condition goes 'X', an assertion message is issued and output is driven to 'X'. Simulation is affected and can fail.

mode=trap — if any condition goes 'X', then only assertion message is issued. Simulation is unaffected.

mode=none — excludes module or instance for X propagation. Object field is required to specify the particular module or instance.

report=fatal|error|warning|info|suppress — applies checking only on the severity level specified (error is default).

object=<hier_path> | <du>[/<inst>] | <hierlevel>

'+' separates the different object selections. '.' is used to indicate recursion. '*' a wildcard character for selecting design units (for example: -xprop, object=du* selects all design units whose names begin with du). Regular expressions are supported for module searching. For example:

```
vopt -xprop,object=/top/u1+/top/u2+/top/u3.+du1. \
-xprop,mode=none,object=top/u3.inst1
vopt -xprop
```

This example includes: u1, u2, all of u3 recursively (exclusive of inst1), and all of du1 recursively.

<hierlevel> is a positive integer that sets the number of levels of hierarchy included as part of processing by the -xprop argument. For example, for the following hierarchy:

```
-xprop,object=/tb/topA+2
selects "topA" and "midA" instances only, where:
tb --> topA --> midA --> botA
|
--> topB --> midB --> botB
```

For more information on applying X propagation, refer to [X Propagation in Simulation](#) in the *Questa User's Manual*.

- **-xprop_disable=stduchcheck**

(optional) For VHDL instances in a design, this argument disables global checking for std_logic U values as part of managing X-propagation.

- **-xprop_enable=[glitchfreeassert] [indexcheck] [optmode]**

Enables the following actions of managing X-propagation on entire design:

glitchfreeassert — Remove noise in assertions resulting from glitches.

indexcheck — performs X-propagation checking only on out-of-bounds arrays and array indexes—no other xprop functionality is applied to the design.

optmode — removes an unreachable branch so that the -xprop argument ignores output values from unreachable (dead or optimized) branches.

- **<design_unit>**

One or more top-level design units that you want to optimize. Required. You can use package names; they are 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 (_). You cannot use this vopt option if you use the 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 of 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 while specifying an exclude file:

```
vopt top +acc -pa_upf test.upf -pa_excludefile bar_exclude -o top_out
```

```
#Contents of bar_exclude:
module_name /interleaver_tester/dut/"mem_inst[1-4]" -s "q.**"
```

- Enable the display of Start, End, and Elapsed time as well as a message count summary. Disables echoing of the command line.

```
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 VHDL instance (u1) and below:

```
vopt -assertdebug+/top/u1.
```

- Specify Liberty files and create a Liberty database:

```
vopt -pa_libertyfiles=test1/a.lib,test2/b.lib,test3/c.lib -pa_dumplibertydb=result/db
```

- Load a Liberty database:

```
vopt -pa_upf test.upf -o design_opt tb -pa_genrpt=conn+cell  
-pa_loadlibertydb=result/db
```

Related Topics

[Optimizing Designs with vopt \[Questa SIM User's Manual\]](#)

[Preoptimizing Regions of Your Design \[Questa SIM User's Manual\]](#)

vsim

Invokes the VSIM simulator. Invoking this command with the `-view` argument enables you to view the results of a previous simulation run.

Syntax

This section lists all arguments of the `vsim` command in alphabetical order.

The Arguments section groups the argument descriptions into the following categories:

Note

Argument Groups

- [All languages](#)
 - [VHDL Arguments](#)
 - [Verilog Arguments](#)
 - [SystemC Arguments](#)
 - [Object Arguments](#)
-

`vsim [arguments]`

[arguments]:

[[-32](#) | [-64](#)]

[[-allowcheckpointcpp 1|0](#)] [[-appendlog](#)] [[-assertcounts](#)] [[-accessobjdebug](#) | [-noaccessobjdebug](#)]
[[-assertcounts](#) | [-noassertcounts](#)] [[-assertdebug](#) | [-noassertdebug](#)] [[+alt_path_delays](#)] [[-assertfile <filename>](#)] [[-assume](#) | [-noassume](#)] [[-autoexclusionsenable=<exclusion_type>](#)]
[[-autoprofile\[=<profile_database>\]](#)] [[-autofindloop](#)]
[[-batch](#)] [[+bitblast\[=<iopath | tcheck>\]](#)]
[[-c](#)] [[-capacity\[=<line>\]](#)] [[-capstats\[=<args>\[,\]\]](#)] [[-checkvifacedrivers](#)] [[-classdebug](#) | -noclassdebug]
[-codelink=<home_path> | [-nocodelink](#)] [[-colormap new](#)] [[-compress_elab](#)]
[-coverage] [-coveranalysis] [[-coverenhanced](#)] [[-coverstore <dir_path>](#) -testname <name>]-ovmtestname|-uvmtestname [[-multicount\[=<a|b|c|d|f|g|s|t>\]](#)] [[-seed](#)] [[-cppinstall](#)
<[gcc | g++] version>] [[-cpppath <filename>](#)] [[-cvgbintstamp](#)] [[-cvgcollapseembeddedinstances](#)] [[-cvghaltillbin](#)] [[-cvgmaxrptrhscross](#)] [[-cvgmergeinstances](#)
| [-nocvgmergeinstances](#)] [[-cvgopt\[=<mode>\[,<mode>...\]\]](#)] [[-cvgperinstance](#)] [[-cvgprecollect <ucdb_file>](#)] [[-cvgsingledefaultbin](#)] [[-cvgzwnocollect <1 | 0>](#)]
[-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] [-enabledpisoscb] [-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] [+initmem+<seed>] [+initreg+<seed>] [-initreport <filename>] [+initregNBA | +noinitregNBA] [-installcolormap] [+int_delays] [-iterationlimit[=<limit>[k][m][g]] | [i[ncrease]]]
 [-keeploaded] [-keeploadedrestart] [-keepstdout]
 [-logfile <filename> | -l <filename> | -nolog] [-L <library_name> ...] [-ldflags <"linkopts">] [-lib <libname>] [-libverbose[=plib]] [<library_name>.<design_unit>] [-learn <root_file_name>] [<license_option>] [-Ldir <pathname> [<pathname> ...]] [-Lf <library_name> ...] [-L mtiPA] [-load_elab <filename>]
 [+maxdelays] [+mindelays] [-mlopt] [-modelsimini <path/modelsim.ini>] [-msgfile <filename>] [-msglimit {error | warning | all[,-<msgNumber>,...] | none[,+<msgNumber>,...] | <msgNumber>,[msgNumber,...]}] [-msglimitcount <limit_value> -msglimit [all, | none,] <msgNumber>[, <msgNumber>...]] [-msglimitcount <limit_value> -msglimit [error|warning]] [-msgmode both | tran | wlf] [-multisource_delay min | max | latest] [+multisource_int_delays] [-mvchome <path>]
 [-name <name>] [-noappendclose] [+no_autodtc] [-noautoldlibpath] [-nodpiexports] [-no_autoacc] [-noautofindloop] [+no_cancelled_e_msg] [+no_glitch_msg] [-noimmedassert] [+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] [-noexcludetary] [-nofileshare] [-noimmedca] [-noltop] [-noimplicitcoverpoint] [-noglitch] [-noscmainscopename] [-nopsl] [+nosdferror] [+nosdfwarn] [+nospecify] [-nostdout] [-nosva] [-note <msg_number>[,<msg_number>,...]] [+notifier_ondetect] [+notiftoggle01[+<seed>]] [+notimingchecks | +ntcnotchks] [-notogglealias] [-notoggleints] [-notogglevhdlrecords] [-noverboseprofile] [-novhdlvariablelogging] [+nowarnBSOB] [+nowarn<CODE | number>] [-nowiremodelforce] [+ntc_warn] [+ntcnotchks]
 [-oldvhdlforgennames] [-onElabError [<value>]] [-onfinish ask | stop | exit | final] [-optionset <optionset_name>] [-OVMdebug] [+OVM_TESTNAME=<testname>]
 [-pa] [-pa_allowtimezeroevent[=all]] [-pa_debugdir <directory>] [-pa_disabletimezeroevent] [-pa_gls <testbench_top>] [-pa_highlight] [-pa_lib <pathname>] [-pa_loadimdb] [-pa_togglelimit=<integer>] [-pduiignore[=<instpath>]] [-pdupath[=<lib_path>]] [-pedanticerrors] [-permissive] [-permit_unmatched_virtual_intf] [-pli "<object list>"] [-plicompatdefault [latest | 2009 | 2005 | 2001]] [+<plusarg>] [-postsimdataflow] [-postsimdataflow]

```
printforces] [-printsimstats[=<val>][v]] [-psloneattempt] [-pslinfinitethreshold=<integer>]
[+pulse_e/<percent>] [+pulse_e_style_ondetect] [+pulse_e_style_onevent] [+pulse_r/
<percent>] [+pulse_int_e/<percent>] [+pulse_int_r/<percent>]

[-quiet] [+questa_mvc_core+wlf_disable <filename>] [-qwavedb[=+<option>[+<option>+...]]]
[-restore <filename>] [-runinit]

[-sc22] [-noscmainscopename] [-scchkpntrestore] [-scdpidebug] [-sclib <library>] [-scstacksize
<value>] [-sc_arg <string> ...] [+sdf_iopath_to_prim_ok] [-sdfallowvlogescapeport]
[+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>] [-solvefailseverity=<value1>[<value2>]]
[-solvefailtestcase[=<filename>]] [-solveprofile] [-solvrev <version>] [-solvetimeout
<val>] [-solveverbose] [-stats [=+ | -]<feature>[,[+ | -]<mode>]] [-strictvital] [-suppress
{<msgNumber> | <msgGroup>} , [<msg_number> | <msgGroup>,...]] [-sv_lib
<shared_obj>] [-sv_liblist <filename>] [-sv_root <dirname>] [-sv_seed <integer> | random]
[-svext=[+|-]<extension>[,[+|-]<extension>]... ] [-svrandext=[+|-]<extension>[,[+|-]
]<extension>]... ] [-sync] [-syncio | -nosyncio]

[-t [<multiplier>]<time_unit>] [-tab <tabfile>] [-tag <string>] [-title <title>] [-togglecountlimit
<int>] [-togglemaxintvalues <int>] [-togglemaxrealvalues <int>] [-togglepackedasvec] [-
toggleportsonly] [-togglevlogenumbits] [-togglevlogreal] [-togglewidthlimit <int>] [-
trace_dpi <val>] [-trace_foreign <int>] [+transport_int_delays] [+transport_path_delays]
[+typdelays]

[-ucdbteststatusmsgfilter <TCL_subtext>] [-undefsyms={<args>}] [-
usenonstdcoveragesavesysf] [-uvmcontrol={<args>}] [+UVM_TESTNAME=<testname>]

[-v2k_int_delays] [-vcdread <filename>] [-vcdstim [<instance>=<filename>] [-
verboseprofile][-version] [-vhdlmergedupackage] [-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] [-
voptargs=<args>] [-vopt_verbose] [-vpicompatcb] [-vv]

[-warning <msg_number>[,<msg_number>,...]] [-warning error] [-wlf <file_name>] [-
wlfcachesize <n>] [-wlfcollapsedelta] [-wlfcollapsetime] [-nowlfcollapse] [-wlfcollapse]
[-nowlfcollapse] [-wlfdelteonquit] [-nowlfdelteonquit] [-wllock] [-nowllock] [-wlfopt]
[-nowlfopt] [-wlfsimcachesize <n>] [-wlfslim <size>] [-wlftlim <duration>] [-work
<pathname>] [-wrealdefaultzero] [-wreal_resolution <val>]
```

Description

You can also view coverage data stored in a UCDB or a coverstore from a previous simulation run (when invoked with the `-viewcov` argument). 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, from the Transcript window of the Main window, or from the GUI (select Simulate > Start Simulation).

Package names can be used at the command line and are treated as top-level design units.

All arguments to the `vsim` command are case-sensitive; for example, `-g` and `-G` are not equivalent.

Arguments

All languages

- `-32 | -64`

(optional) Specifies whether `vsim` uses the 32-bit or 64-bit executable, where 32-bit is the default.

These arguments apply only to the supported Linux operating systems.

These arguments override the `MTI_VCO_MODE` environment variable, which applies only to executables used from the `<install_dir>/bin/` directory. Therefore, these arguments are ignored if you run `vsim` from an `<install_dir>/<platform>/` directory.

You can specify these arguments only on the command line; they are not recognized as part of a file used with the `-f` argument.

- `-allowcheckpointcpp 1|0`

(optional) Enable/disable support for checkpointing foreign C++ libraries. Must be used in the same `vsim` session in which the checkpoint is created. Valid arguments are 1 and 0.

1 : turn on the support

0 : turn off the support

This argument is not supported on Windows platforms.

- `-appendlog`

(optional) Append simulation data to the default log file, or the log file created with the `-logfile` argument.

- **-assertcounts**

(optional) Enables extended count information, including: pass count, vacuous pass count, disabled count, attempted count. Passes enabled counts to coverage reports during both live simulation (coverage report) and post-processing (vcover report) modes. You can change the default behavior by setting the AssertionCover variable in the *modelsim.ini* file. Refer to [AssertionCover](#) in the User's Manual.

Note

 The vsim -assertcounts and -noassertcounts arguments were formerly named -assertcover and -noassertcover. The -assertcover and -noassertcover arguments are still supported, but their use is deprecated.

The following stipulations apply to the reported assertion counts:

- At simulation time, if you have specified -assertcounts, the pass count for each assertion is a 32-bit bin which records the actual count of assertion passes. Assertion Successes are those assertions that have never failed and have passed at least once non-vacuously.
 - If you have not specified either -assertcounts or -assertdebug, the pass count for each assertion is only a one-bit bin. This bin is set to 1 if the assertion passes and 0 if the assertion status is anything else (fail, vacuous, disable, or active). Assertion Successes are only those assertions that have never failed and have passed at least once non-vacuously.
 - Assertion Attempts are the sum of all evaluated assertions.
- **-assertdebug**

(optional) Requires vopt +acc=a, vopt -assertdebug, or vsim -voptargs="-assertdebug". If assertions are logged, the -assertdebug argument stores assertion data in the WLF file so you can analyze assertion passes and failures in the assertion debug pane of the Wave window. This argument 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 want to see only this information, you should specify -assertcounts instead.
- Causality traceback feature.
- Reporting of assertion pass messages for PSL only (disabled by default).

You can change the default behavior 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 [AssertionDebug](#) and “[Analyzing Assertion Failures in the Assertion Debug Pane of the Wave Window](#)” in the User’s Manual for more information.

- **-assertfile <filename>**

(optional) Designates an alternative file for recording VHDL/PSL/SVA assertion messages.

You can also specify an alternate file with the AssertFile *modelsim.ini* variable. By default, assertion messages are output to the file specified by the TranscriptFile *modelsim.ini* variable. Refer to [AssertFile](#), [TranscriptFile](#), and “[Creating a Transcript File](#)” in the User’s Manual for more information.

- **-assume**

(optional) Simulates PSL and SystemVerilog assume directives as though they were assert directives. You can also specify this argument with the SimulateAssumeDirectives variable in the *modelsim.ini* file. Refer to [SimulateAssumeDirectives](#) and “[Processing Assume Directives](#)” in the User’s Manual for more information.

- **-autoexclusionsdisable=<exclusion_type>**

(optional) Disables automatic code coverage exclusions for:

- FSMs and its 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 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 (refer to [AutoExclusionsDisable](#) in the User’s Manual). If an FSM state is excluded, all transitions from and to this state are also excluded.

- **-autoprofile[=<profile_database>]**

(optional) Causes vsim to automatically collect performance profile data without requiring the use of profile commands. This argument also enables the enhanced routines. (Does not collect memory profile data.) You can specify a profile database name (optional):

[=<profile_database>] — If you do not specify <profile_database>, then a name is created from the date, time and process PID.

Note

 Use of -autoprofile disables the following profile commands: profile interval, profile clear, profile on, profile off, profile reload, and profile open.

Refer to the individual profile commands reference entries and “[Profiling Performance and Memory Use](#)” in the *Questa SIM User’s Manual* for more information.

- **-autofindloop**

Attempts to find a zero delay loop and define it in a textual report when a simulation exceeds the iteration limit. The report identifies active processes along with event and signal activity, when possible. Reported results can vary, depending on design content, optimization level, and design visibility. Enabled by default, but can be disabled with **-noautofindloop**. For nonGLS designs, add **vopt +acc=l** to enhance the zero delay loop report.

- **-batch**

(optional) Runs scripted batch simulations using the **-do** argument to **vsim**. Must be specified from a Windows command prompt or a UNIX terminal. The simulator returns an error if you use **-batch** 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 **-c**, **-gui**, or **-i** are not used. Refer to [BatchMode](#) in the User’s Manual.

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 (refer to [TranscriptFile](#) and [BatchTranscriptFile](#) in the User’s Manual.) You can also disable sending transcript data to **stdout** by specifying **vsim -nostdout**, but you must then save transcript data to a file. Refer to “[Batch Mode Simulation](#)” in the User’s Manual 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, this argument 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 argument is intended for use with applications employing SDF annotation.

- **-c**

(optional) Specifies to run the simulator in command-line mode. Refer to “[General Modes of Operation](#)” in the User’s Manual for more information.

- **-capacity[=line]**

(optional) Enables the fine-grain analysis display of memory capacity. (The default is a coarse-grain analysis display.) The “**=line**” option allows generation of the point of allocation along with the point of declaration.

-
- `-capstats[=<args>[,]]`

(optional) Generates the capstats report. The possible args are:

`du [+summary | +details | +duname=<duname>] | decl | line | eor | filename=<filename>`

Specifying no arguments generates a summary report at the end of the simulation.

To generate a more detailed report for specific types, specify a single argument or a comma-separated list of arguments.

- `du` — generates a report that includes the memory usage of each design unit. You must also specify the `-capacity` argument with `vsim`.
 - The `+summary` argument provides a summary report of the memory usage within a design unit.
 - The `+details` argument provides a more expansive report that includes details about dynamic allocations.
 - The `+duname=<duname>` argument generates a report that covers only the specified design unit, and you can enter the argument multiple times to specify multiple design units.
- `decl` — generates a declaration-based report for SV dynamic objects. You must also specify the `-capacity` argument with `vsim`.
- The `line` argument generates a line-based report for SV dynamic objects. You must also specify the `-capacity=line` argument with `vsim`.
- `eor` — generates a summary report at the end of each run command.

`filename=<filename>` — prints the report to the specified file. If you do not specify a filename, the report prints in the transcript file.

- `-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.

You can use these arguments only when first invoking the `vsim` command, and not later from a `.do` script.

Note

 You cannot use the `-codelink=` argument at the same time as an explicit `-foreign codelink` argument. Use only one or the other — never both at the same time.

- **-colormap new**
(optional) Specifies that the window 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](#)” in the User’s Manual for more information.
- **-coverage**
(optional) Enables code coverage statistics collection during simulation.
Important: You must use the +cover argument during compilation or optimization in order for coverage to be collected and displayed.
- **-coveranalysis**
(optional) Display alias toggle nodes in coverage reports and in the GUI, and save those nodes in the UCDB files.
- **-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>|-ovmtestname|-uvmtestname [-multicount[=-a|b|c|-d|f|-g|s|t]] [-seed]**
(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 the specification of -testname <name>, -ovmtestname, or -uvmtestname, 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 that use -coverstore must use the same directory path, and their design hierarchy must be identical.
 - testname <name>**
Name of the test whose coverage is being saved.
 - ovmtestname**
Specifies that the +OVM_TESTNAME name is used both for the testnames and for the names of the data files produced.
 - uvmtestname**

Specifies that the +UVM_TESTNAME name is used both for the testnames and for the names of the data files produced.

-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 change them to single-bit with the -a, -d, or -g modifiers. You can use multiple modifiers; if you do not specify any modifiers, 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.

-seed

(optional) When used in conjunction with a simulation run using vsim -sv_seed, this argument appends "_<seed_value>" to the saved ovmtestname or uvmtestname. It is not compatible with -testname. Only supported during live simulation, not in Coverage View mode. If no seed value has been set, zero (0) is applied.

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
```

When -dpicppinstall or -dpicpppath arguments are also present, the order of precedence in determining the compiler path is the following:

- The -dpicppinstall argument is used with vsim
- The -dpicpppath argument is used with vsim
- The -cppinstall argument is used with vsim
- The -cpppath argument is used with vsim

Refer to “[Supported Platforms and Compiler Versions](#)” in the User’s Manual 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. Refer to [CppPath](#) in the User's Manual.

When -dpicppinstall or -dpicpppath arguments are also present, the order of precedence in determining the compiler path is the following:

- The -dpicppinstall argument is used with vsim
- The -dpicpppath argument is used with vsim
- The -cppinstall argument is used with vsim
- The -cpppath argument is used with vsim

- **-debugdb=<db_pathname>**

(optional) Instructs Questa SIM to generate a database of connectivity information to use for post-sim debug in the Dataflow and Schematic windows. The database pathname should have a .*dbg* extension. If you do not specify a database pathname, Questa SIM creates a database file named *vsim.dbg* in the current directory.

Questa SIM reuses the existing .*dbg* file, and prints a note to the transcript, when you specify -debugdb for a design that has not changed since the creation of the database.

Refer to “[Post-Simulation Debug Flow Details](#)” in the User's Manual for more information.

- **-default_radix <radix>**

(optional) Sets the default radix for the simulation and overrides the DefaultRadix preference variable. <radix> can be any of the following: ascii, binary, decimal, hexadecimal, octal, symbolic, unsigned. Refer to [DefaultRadix](#) in the User's Manual for more information.

- **-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 “[Message Viewer Window](#)” in the User's Manual for more information.

-
- both — outputs messages to both the transcript and the WLF file.
 - tran — outputs messages only to the transcript; messages are not available in the Message Viewer. Default behavior.
 - wlf — outputs messages only to the WLF file/Message Viewer; messages are not available in the transcript.

The 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. You can specify multiple commands 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 become the following script:

```
"force clk 0 0, 1 10 -r 20; do testfile.do; run -all"
```

You can include nested vsim-do operations. A vsim command do-file that contains another vsim command with its own do-file executes the nested do-file.

- **-donotcollapsepartiallydriven**
(optional) Prevents the collapse of partially driven and undriven output ports during optimization. Prevents incorrect values that can occur if the output ports collapse.
- **+dumpports+collapse | +dumpports+nocollapse**
(optional) Determines whether vectors (VCD id entries) in dumpports output are collapsed or not. The default behavior is collapsed, but you can change the default behavior by setting the DumphportsCollapse variable in the *modelsim.ini* file. Refer to [DumphportsCollapse](#) in the User's Manual.
- **+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](#)” in the User's Manual for additional information.

- **+dumpports+unique**
(optional) Generates unique VCD variable names for ports in a VCD file, even if those ports connect to the same collapsed net.
- **-elab <filename>**
(optional) Creates an elaboration file for use with -load_elab. Supports automatic collection of performance profile data with -autoprofile. Refer to “[Simulating with an Elaboration File](#)” in the User’s Manual for more information.
- **-elab_cont <filename>**
(optional) Creates an elaboration file for use with -load_elab and then continues the simulation. Supports automatic collection of performance profile data with -autoprofile.
- **-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 do not support checkpoint/restore. Note that FLI models sensitive to design load ordering may still not work correctly even if you use this argument.
- **-enabledpisoscb**
(optional) Enables DPI export calls from the SystemC *start_of_simualtion()* callback. A side effect of using this argument is that SystemC signal writes done in *start_of_simulation()* callback do not reflect the updated value at time 0. Insert a delta cycle using a wait statement in the processes to update these signals to the correct value. Also, with mixed language simulation, process execution order may change at time 0 with this argument. This argument overrides the value of the EnableDpiSosCb *modelsim.ini* variable. Refer to [EnableDpiSosCb](#) in the User’s Manual.
- **-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](#)” in the User’s Manual 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. Refer to [ExtendedToggleMode](#) in the User's Manual for more information.

- **-f <filename>**
(optional) Specifies a file with more vsim command arguments. Enables you to reuse complex argument strings without retyping.
Refer to "[“Argument Files”](#) on page 37" 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”](#) in the User's Manual for more information.
- **-feccountlimit [<n> | 0]**
(optional) Limits the number of counts that are tracked for Focused Expression Coverage. When a bin reaches the specified count, coverage ignores further tracking of the inputs linked to the bin.

Note

 Changing this value from the default can 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”](#) in the User's Manual 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 "[“Viewing FSM Information in the GUI”](#) in the User's Manual 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 from defparams (such as top-level generics/parameters and generics/parameters that would otherwise receive their default values).

You can enter multiple -g arguments, one for each generic/parameter, 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. You can prefix the name with a relative or an 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 a value for the declared data type of a VHDL generic, or any legal value for a Verilog parameter. Any value you specify for a VHDL generic must be 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 argument 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, you cannot set generics/parameters of composite type (arrays and records) 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, add 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 in escaped quotes (\\"), for example:

```
-gstrgen=\\"This is a string\\\"
```

- -G<Name>=<Value> ...

(optional) Same as -g (see above) except that -G also overrides generics/parameters that received explicit values in generic maps, instantiations, or from defparams.

This argument provides 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. You can prefix the name 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. The value you specify for a VHDL generic must be

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 for other shared objects to reference and use. If you specify multiple, comma-separated, shared objects, they are merged internally and then loaded as a single shared object.

If any of the shared objects have circular dependencies, you must enclose all the arguments in quotation marks (""), for example:

```
-gblso "lib1.so, lib2.so"
```

You can also specify this argument with the GlobalSharedObjectsList variable in the *modelsim.ini* file. Refer to [GlobalSharedObjectsList](#) in the User's Manual.

- **-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.

- **+initregNBA | +noinitregNBA**

(optional) Controls whether +initreg settings applied to registers of sequential UDPs are non-blocking. This is useful when continuous assignments overwrite register initialization.

+initregNBA — (default) enables this functionality.

+noinitregNBA — disables this functionality.

- **-installcolormap**

(optional) For Linux only. Instructs vsim to use its own colormap for the display.

- **+int_delays**

(optional) Optimizes annotation of interconnect delays. This argument causes optimized cells to preallocate/initialize memory for delay elements (SIPD) that are used to implement interconnect delays to cell input ports. The default is to allocate/initialize this memory when the cell is annotated from SDF. Using +int_delays allocates the delay element with a zero delay.

- **-iterationlimit[=<limit>[k][m][g]] | [i[ncrease]]**

(optional) Sets the iteration limit; or, enables a stepwise increase of the iteration limit as needed. The iteration limit specifies the number of simulation kernel iterations to allow before advancing time. The default iteration limit is 10 million, set by the IterationLimit variable of the *modelsim.ini* file. Refer to [IterationLimit](#) in the User's Manual.

=<limit>[k][m][g]

<limit> – An unsigned integer value that overrides the iteration limit set by the IterationLimit variable. Use the “set IterationLimit <value>” command to override this value.

k – Designates units in thousands.

m – Designates units in millions.

g – Designates units in billions.

=i[ncrease]

Sets the simulator to automatically increase the iteration limit—if the default iteration limit of 10 million is reached—and report the final setting at the end of simulation. If the simulator reaches the default limit of 10 million iterations, it issues a warning about a possible zero delay loop, and sets a flag to report the final iteration limit at the end of simulation. Each time the iteration exceeds the new limit, the value is increased by 500 thousand. The value that the simulator reports at the end of simulation is the final iteration limit that was set, not the maximum simulator iterations that were performed. The maximum allowed iteration limit value 4,294,967,000.

- **-keeploaded**

(optional) Prevents the simulator from unloading/reloading any HDL interface shared libraries when it restarts or loads a new design. The shared libraries remain loaded at their current positions. User application code in the shared libraries must reset its internal state during a restart for this to work effectively.

- **-keeploadedrestart**

(optional) Prevents the simulator from unloading/reloading any HDL interface shared libraries during a restart. The shared libraries remain loaded at their current positions. User application code in the shared libraries must reset its internal state during a restart for this to work effectively.

Use this argument if you will be doing warm restores after a restart, and the user application code has set callbacks in the simulator. If you do not use this argument, and the shared library is loaded into a new location, the callback function pointers might not be valid.

- **-keepstdout**

(optional) For use with foreign programs. Instructs the simulator not to 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>`. You can use the abbreviation `-l <filename>`. Overrides the default transcript file creation set with the `TranscriptFile` or `BatchTranscriptFile` `modelsim.ini` variables. You can also specify “`stdout`” or “`stderr`” for `<filename>`. Refer to [TranscriptFile](#) or [BatchTranscriptFile](#) in the User’s Manual.

`-nolog` — Disables transcript file creation. Overrides the `TranscriptFile` or `BatchTranscriptFile` variables set in the `modelsim.ini` file.

Use `-appendlog` to append simulation data to the log file.

Refer to “[Batch Mode](#)” in the User’s Manual for more information about saving transcript data.

- `-L <library_name> ...`

(optional) Specifies the library to search for top level design units instantiated from Verilog, and for VHDL default component binding. Prints a list of all visible top level libraries, if a top level design unit cannot be found. Refer to “[Library Usage](#)” in the User’s Manual for more information. If you specify multiple libraries, each must be preceded by the `-L` argument. Libraries are searched in the order in which they appear on the command line.

- `-ldflags <"linkopts">`

(optional) Specifies (in quotation marks) any options for linking auto compiled DPI object files. Supports multiple occurrences.

- `-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. Generates the following files:

```
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.

Tip

Specifying this argument with `-voptargs=+acc` enables as much visibility as possible to help ensure functional correctness of your design.

- `-Ldir <pathname> [<pathname> ...]`

(optional) Passes one or more container folders for libraries specified by either `vsim -L` or `vsim -Lf`. Once you specify a container folder (`pathname`), you can directly reference the libraries in this folder using their logical names. You can specify multiple values for

pathname. Questa SIM searches multiple paths in the order in which you specify them on the command line.

Note

 Questa SIM searches the current working directory (\$cwd) before any pathnames you specify for -Ldir, as if there was an implicit vsim -Ldir . specified first on the command line.

- **-Lf <library_name> ...**
(optional) Same as -L but libraries are searched before `uselib directives. Refer to “[Library Usage](#)” in the User’s Manual for more information.
- **-L mtiPA**
(optional) Use library of precompiled Power Aware behavioral models.
- **-lib <libname>**
(optional) Specifies the working library in which vsim will look for the design unit(s). Default is "work".
- **-libverbose[=prlib]**
(optional) Enables verbose messaging about library search and resolution operations. The “=prlib” option prints out the -L or -Lf argument that was used to locate each design unit loaded by vopt or vsim. This information is printed to the right of the existing “Loading design unit xyz...” messages.
Libraries containing top design units that are not explicitly present in the set of -L/-Lf arguments are implicitly promoted to searchable libraries at the end of the library search order. They appear as -Ltop in the output of the -libverbose argument. To stop creation of -Ltop libraries, use the -noltop argument of vopt or vsim.
- **<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 you invoke 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.

<licen se_opt ion>	Description
-	check out
lic_lnl	msimhdlsim license
_only	only

<licen se_opt ion>	Description
- check out	
lic_mi msimhdlsmi/	
xed_o msimhdmlmix licenses	
nly only	
- exclude msimhdlsmi	
lic_no license	
_lnl	
- exclude msimhdmlmix	
lic_no license	
_mix	
- exclude qhsimvh	
lic_no license	
_slvhd	
l	
- exclude qhsimvl	
lic_no license	
_slvlo	
g	
- do not wait in queue	
lic_no when license is	
queue unavailable	
- check out PLUS	
lic_plu (VHDL and Verilog)	
s ¹ license immediately	
after invocation	
- check out VHDL	
lic_vh license immediately	
dl ¹ after invocation	
- check out VLOG	
lic_vlo license immediately	
g ¹ 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 (refer to [License](#) in the User's Manual). As with the command line arguments, these settings affect only the initial invocation of vsim and do 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`. Supports automatic collection of performance profile data with `-autoprofile`. Refer to “[Simulating with an Elaboration File](#)” in the User’s Manual for more information.

- `-mlopt`

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 situations, differences in simulation results can occur. The differences primarily relate to initial values on mixed-language nets. The most common difference is VHDL nets that may have an initial value of 'X' instead of 'U'.

Limitations:

- 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 argument 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 argument is ignored if the design has analog components, when doing power aware simulation.
- `-modelsimini <path/modelsim.ini>`

(optional) Loads an alternate initialization file that replaces the current initialization file. Overrides the file path specified for the MODELSIM environment variable. Specifies either an absolute or relative path to the initialization file, including the filename. On Windows systems, use a forward slash (/) for the path separator character.

Restriction

 Do not use this argument in a file specified with the `vsim -f` argument. Questa SIM will display an error message.

- `-msgfile <filename>`

(optional) Designates an alternative file for recording error messages. You can also specify an alternate file with the Error File `modelsim.ini` variable. By default, error messages are

output to the file specified by the TranscriptFile variable in the *modelsim.ini* file. Refer to [ErrorFile](#) and [TranscriptFile](#) in the User's Manual.

- **-msglimit** {error | warning | all[,-<msgNumber>,...] | none[,+<msgNumber>,...] | <msgNumber>,[msgNumber,...]}

(optional) Limits messages to the default message limit count of five. Use the -msglimitcount argument to change the default message limit count. Specify multiple messages as a comma-separated list.

error — Stops the run when the total number of errors reaches the default count.

warning — Stops the run when the total number of warnings reaches the default count.

all — Limits all messages to the default count except those you specifically exclude. To exclude messages from the limit, supply a comma-separated list of message numbers, each preceded by a minus sign (-), for example “all,-<msgNumber1>,-<msgNumber2>,...”.

none — Excludes all messages from the default count except those you specifically include. To include messages to limit to the default count, supply a comma-separated list of message numbers, each preceded by a plus sign (+), for example “none,+<msgNumber1>,+<msgNumber2>,...”.

<msgNumber>[,<msgNumber>,...] — Specifies messages to limit to the default count.

Examples:

- Stop the run when the total number of errors reaches the default count:

```
-msglimit error
```

- Stop the run when the total number of warnings reaches the default count:

```
-msglimit warning
```

- Specifically limit messages 2374 and 2385 to the default count:

```
-msglimit 2374,2385
```

- Limit all messages to the default count, except messages 2374 and 2385:

```
-msglimit all,-2374,-2385
```

- Limit only messages 2374 and 2385 to the default count:

```
-msglimit none,+2374,+2385
```

- **-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. Refer to [MsgLimitCount](#) in the User's Manual.

- **-msglimitcount <limit_value>** -msglimit [error|warning]
(optional) Stops the run when the total number of reported errors/warnings reaches the user-defined limit_value.
 - **-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 window.
 - wlf — Outputs messages only to the WLF file/Message Viewer and Results Analysis windows, therefore they are not available in the transcript.
- Refer to "[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 can 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 arguments (described below).

You cannot use the +multisource_int_delays argument 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. Refer to [MvcHome](#) in the User's Manual.
 - **-name <name>**
(optional) Specifies the application name used by the interpreter for send commands. This does not affect the title of the window.

-
- **-noassertcounts**

(optional) Disables extended count information for assertions. You can also specify this argument with the AssertionCover variable in the *modelsim.ini* file. Refer to [AssertionCover](#) in the User's Manual.

Note

 The vsim -assertcounts and -noassertcounts arguments were formerly named -assertcover and -noassertcover. The -assertcover and -noassertcover arguments are still supported, but their use is deprecated.

- **-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. You can also specify this argument with the AssertionDebug variable in the *modelsim.ini* file. Refer to [AssertionDebug](#) in the User's Manual.

- **-noassume**

(optional) Disables simulation of PSL and SystemVerilog assume directives. You can also specify this argument with the SimulateAssumeDirectives variable in the *modelsim.ini* file. Refer to [SimulateAssumeDirectives](#) and “[Processing Assume Directives](#)” for more information.

- **-no_autoacc**

(optional) Prevents vsim from automatically passing the +acc argument to vopt. Specifying this argument prevents vopt from opening any Verilog PLI modules for accessibility. You can pass specific +acc options to vopt by using the -voptargs argument.

- **-noautofindloop**

(optional) Disables -autofindloop.

- **-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. You can also specify this argument with the CheckpointCompressMode variable in the *modelsim.ini* file. Refer to [CheckpointCompressMode](#) in the User's Manual.

- **-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 are always skipped.
- **-noimmedassert**
(optional) Controls the simulation of immediate assertions (Verilog and VHDL). By default, immediate assertions are simulated. Use the **-noimmedassert** argument to disable simulation of immediate assertions. 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. Refer to [SimulateImmedAsserts](#) in the User's Manual.
- **-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 can also set even ordering with the **ImmediateContinuousAssign** variable in the *modelsim.ini* file. Refer to [ImmediateContinuousAssign](#) in the User's Manual.
- **-noltop**
Stops creation of -Ltop libraries. Libraries containing top design units that are not explicitly present in the set of -L/-Lf arguments are implicitly promoted to searchable libraries, and placed at the end of the library search order. They appear as -Ltop in the output of the -libverbose argument unless you use the **-noltop** argument. See the [-libverbose\[=plib\]](#) option.
- **+no_notifier**
(optional) Disables the toggling of the notifier register argument of all timing check system tasks. By default, the notifier toggles 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 remains 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 Simulation](#)” in the User’s Manual 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 [note](#) and “[Message Severity Level](#)” in the User’s Manual for more information.
- **+notifier_ondetect**
(optional) Causes X output values generated by timing check notifier toggles 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. Refer to [Approximating Metastability](#) in the User’s Manual.
- **+notimingchecks | +ntcnatchks**
(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. Does not affect generics with the name TimingChecksOn on non-VITAL models.) By default, Verilog timing check system tasks (\$setup, \$hold,...) in specify blocks are enabled. For VITAL, the ASIC or FPGA vendor controls the timing check default, but most default to enabled.

Additionally, +ntcnatchks maintains the delay net delays that negative timing check limits make necessary. For this reason when using +ntcnatchks it is necessary to SDF annotate all timing check values.

You can use the [tcheck_set](#) command to disable individual checks.

- **-notogglealias**
(optional) Prevents the storage of alias toggle nodes in the UCDB during the coverage save from simulation. By default, coverage save stores alias toggle nodes.
- **-notogglevhdlrecords**
(optional) By default, VHDL records are automatically included in code coverage metrics. The **-notogglevhdlrecords** argument disables the inclusion of VHDL records, and overrides any setting made with the `ToggleVHDLRecords` *modelsim.ini* variable. Refer to [ToggleVHDLRecords](#) in the User's Manual.
- **-nowiremodelforce**
(optional) Restores the [force](#) command to the previous usage model (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 ““[Optionsets](#)” on page 35” 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 ““[OVM-Aware Debug](#)” in the User's Manual.
- **+OVM_TESTNAME=<testname>**
(optional) Specifies the testname to be used for OVM designs.
- **-pa**
(optional) Performs the simulation in Power Aware mode. Refer to ““[Power Aware Simulation](#)” in the *Power Aware Simulation User's Manual*.
- **-pa_allowtimezeroevent[=all]**
(optional) Enables corruption, isolation, or release of signals at time 0 for some or all events. Use the **-pa_disabletimezeroevent** argument with or without the **=all** option:

With the **=all** option — Enables corruption, isolation, or release of signals at time 0 for all named events in Power Aware. Refer to ““[Named Events in Power Aware](#)” in the *Power Aware Simulation User's Manual*.

Without the **=all** option — Enables corruption, isolation, or release of signals at time 0 for the following events: `pa_corrupt_register`, `pa_iso_on`, and `pa_iso_off`. This functionality is on by default, but you can disable it with the **-pa_disabletimezeroevent** argument.
- **-pa_debugdir <directory>**
Specifies the location of post-simulation debug information for Power Aware simulation. Refer to ““[Power Aware Simulation Debug](#)” in the *Power Aware Simulation User's Manual* for more information.

- **-pa_disabletimezeroevent**
(optional) Disables corruption, isolation, or release of signals at time 0 for the following events: pa_corrupt_register, pa_iso_on, and pa_iso_off.
- **-pa_excludedefaultps**
(optional) Excludes DEFAULT_NORMAL and DEFAULT_CORRUPT power states in the coverage report.
- **-pa_gls <testbench_top>**
(optional) Enables gate-level simulation for Power Aware, performed on the top-of-design test bench (testbench_top).

Note

 This argument is deprecated.

- **-pa_highlight**
(optional) Enables visual indication (highlighting) of power states of signals viewed in the Wave window. A highlighted segment of a waveform indicates an interval when the waveform was corrupted, isolated, or biased.

Tip

 To enable highlighting in the Wave window, you must also use -pa_enable=highlight to enable highlighting in the vopt run.

- **-pa_includeundefined**
(optional) Includes UNDEFINED power states in the coverage report.
- **-pa_lib <pathname>**
(optional) Specifies the library location in which to store the Power Aware information from the vopt run.
- **-pa_loadimdb**
(optional) Loads the contents of the Power Aware database that was dumped with the vopt -pa_dumpimdb command. This database is loaded from either the current working directory or from the library location you specified with the vopt -pa_lib command. Loading this database for simulation makes the Power Aware information model available for UPF query commands in the UPF file.
- **-pa_logfile <filename>**
(optional) Redirects the Power Aware messages to <filename>. By default, these messages are displayed in the transcript window.
- **-pa_togglelimit=<integer>**
(optional) Discontinues reporting an error (8906) for each signal that toggles more than the toggle limit (set by <integer>) during power off and issues note (8922) instead. The default

value of <integer> is 5. This argument affects your simulation only if you used -pa_checks=t or -pa_checks=it with your vopt command.

- -pa_top <pathname>

Note

You do not need to use this vsim argument when you run the vopt -pa_defertop command, though using them together is supported for backward compatibility.

(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. You can then use this argument to change Questa SIM Power Aware simulation at runtime without having to rerun vopt on the same DUT.

- -pduignore[=<instpath>]

Ignore Preoptimized Design Unit (black-box). If you do not specify <instpath>, all PDUs found in compiled libraries are ignored. Otherwise, the PDU specified by <instpath> is ignored. You can specify this argument multiple times using different values of <instpath>. Equivalent to the deprecated -ignore_bbox argument.

- -pdupath[=<lib_path>]

(optional) Specifies library path for a preoptimized design unit (PDU) when the library is moved after top level optimized design unit creation. If you do not specify <lib_path>, the library path is the current working directory.

- -permissive

(optional) Allows messages in the LRM group of error messages to be downgraded to a warning.

You can produce a complete list of error messages by entering the following 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.

- -printforces

(optional) Prints forces/releases during the simulation in the following format:

```
Time 0 ns : force /tb/dut/a 0
Time 10 ns : release /tb/dut/a
Time 100 ns : force /tb/dut/a 1
```

- -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. Refer to [PrintSimStats](#) in the User's Manual.

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`.

- **-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, corresponding to multiple attempts made while executing a top level "always/never" property. With this argument, 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 with the `PslOneAttempt` variable in the *modelsim.ini* file. Refer to [PslOneAttempt](#) in the User's Manual.

- **-pslinfinitethreshold=<integer>**

(optional) Specifies the number of clock ticks that represent infinite clock ticks. Affects only the PSL strong operators; `eventually!`, `until!` and `until_!`. At the end of simulation, if an active strong-property has not clocked this number of clock ticks, neither pass nor fail (that is, 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 also turn on this feature with the `PslInfinityThreshold` in the *modelsim.ini* file. Refer to [PslInfinityThreshold](#) in the User's Manual.

- **+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 you do not specify the rejection limit, 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 can be VITAL or Verilog.

A pulse less than the rejection limit is filtered. If you do not specify the error limit with +pulse_int_e, it defaults to the rejection limit.

- -quiet
 - (optional) Disables 'Loading' messages during batch-mode simulation.
- +questa_mvc_core+wlf_disable <filename>
 - (optional) Disables the creation of a WLF file.
- -qwavedb[=+<option>[+<option>+...]]
 - (optional) Creates the waveform database file required to run Visualizer Debug Environment.

+<option>

A selection of options controlling the information in the *qwave.db* file.

Refer to the “*Mentor Visualizer Debug Environment User’s Manual*” for a list of all available <option> values.

Specify arguments as a string without spaces. You can specify -qwavedb and [-load_elab <filename>](#) together on the vsim command line to load a previously generated elaboration file.

Note



Specifying vsim -qwavedb disables WLF file generation and is incompatible with all WLF arguments and commands.

- -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, including the same type of machine and OS, at least the same memory size, and 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 when a corresponding specify path is not found. Refer to “[SDF to Verilog Construct Matching](#)” in the User’s Manual for additional information.

-
- **-sdfallowvlogescapeport**

Enables SystemVerilog language extension related to SDF annotation. Matches port name in RTL (“\<portname>[0]”) with either of the following in SDF file:

- non-escaped port name (“<portname>[0]”)
- escaped port name (“\<portname>[0]”)

Allows for matching bit-blasted scalar port names in RTL with normal non-escaped port names in SDF.

- **-sdfmin | -sdftypr | -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>= — A specific instance for the associated SDF file. Use this when not performing backannotation at the top level.

<sdf_filename> — The file containing the SDF information.

- **-sdfminr | -sdftypr | -sdfmaxr[@<delayScale>] [<instance>=<sdf_filename>**

(optional) Specifies when an instance, optimized with an associated default SDF file, is to be re-annotated with minimum, typical, or maximum timing from the newly specified SDF file. Note that the “r” in the argument name stands for “replace”.

When you use these arguments to specify a replacement SDF file, you can change the following information only from the initial SDF file used during a previous optimization:

- SDF IO path delay values
- SDF Port delay values
- SDF Device delay values
- SDF interconnect delay values
- SDF timing check limit values
- The SDF min/typ/max selection

You cannot add or delete any statements between SDF versions.

Note

 The simulator assumes that the instance/timing object hierarchy in the new SDF file is compatible with the SDF file specified with the previous optimization.

@<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>= — A specific instance for the associated SDF file. Use this when not performing back-annotation at the top level.

<sdf_filename> — The file containing the SDF information.

The following is a simple usage flow:

Assume that the module top contains three instances (u1, u2, and u3) of a design unit named pduMod.

```
vlib work  
vlog pduMod.v
```

Optimize pduMod and annotate with a generic SDF file *sdf1*.

```
vopt -pdu pduMod -o pduMod_opt -sdfmin pduMod=sdf1  
vlog top.v
```

At simulation, use the default SDF file *sdf1* for the design unit 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 generate before terminating vsim. <n> is the maximum number of missing instance error messages. The default number is 5.
- **-sdfnoerror**
(optional) Changes SDF errors to warnings so that the simulation can continue. Errors issued by the SDF annotator while loading the design prevent the simulation from continuing, whereas warnings do not.
- **-sdfnowarn**
(optional) Disables warnings from the SDF reader. Refer to “[VHDL Simulation](#)” in the User’s Manual 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 “[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. When you also specify the [+multisource_int_delays](#) argument, the **+sdf_verbose** argument adds more detail to the output of the [write timing](#) command.
 - **-stackcheck**
(optional) Enables runtime stack usage sanity checking. This argument enables 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 can lead to a 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. You can obtain detailed **-stats** output by passing the **-stats=*** option to **-voptargs=""** along with the **-vopt_verbose** argument to vsim. Refer to [Two-Step Flow](#) in the User's Manual.

Specify multiple features and modes for each instance of **-stats** as a comma-separated list. You can specify **-stats** multiple times on the command line, but only the final instance takes effect.

You can specify **-printsimstats** and **-stats** on the same command line, however **-stats** always overrides **-printsimstats**, regardless of the order in which you specify the options.

[+ | -] — Controls activation of the feature or mode. The plus character (+) enables the feature, and the minus character (-) disables the feature. You can also enable a feature or mode by specifying it without the plus (+) character. Setting this argument adds or subtracts features and modes from the default settings "cmd,msg,time".

<feature>

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.

Note

 Use -stats=perf+verbose to display information about the operating system and host machine.

<mode>

You can set modes for a specific feature, or globally for all features. To add or subtract a mode for a specific feature, use 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 kilobyte 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](#) in the User's Manual for more information.

- -suppress {<msgNumber> | <msgGroup>} [, [<msg_number> | <msgGroup>] ,...]
(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 [suppress](#) and “[Message Severity Level](#)” in the User's Manual for more information.

<msgNumber>

A specific message number

<msgGroup>

A value identifying a predefined group of messages, based on area of functionality. These groups suppress only notes or warnings. The valid arguments are:

All, GroupNote, GroupWarning, GroupFLI, GroupPLI, GroupSDF, GroupVCD,
GroupVital, GroupWLF, GroupTCHK, GroupPA, GroupLRM

- -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.
 -syncio — (default) I/O is synchronous with simulation activity, always up-to-date.
 -nosyncio — Disables I/O synchronization. This enables vsim to 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> can be 1, 10 or 100.

Do not put a 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:

- VHDL design — Uses the value specified for the Resolution variable in *modelsim.ini*.
- Verilog design with ‘timescale directives — Uses the minimum specified time precision of all directives.
- Verilog design with no ‘timescale directives — Uses the value specified for the Resolution variable in the *modelsim.ini* file. Refer to [Resolution](#) in the User’s Manual.
- Mixed design with VHDL on top — Uses the value specified for the Resolution variable in the *modelsim.ini* file.
- Mixed design with Verilog on top:
 - For Verilog modules not under a VHDL instance — Uses the minimum value specified for their ‘timescale directives.
 - For Verilog modules under a VHDL instance — Ignores all ‘timescale directives (uses the minimum value for ‘timescale directives in all modules not under a VHDL instance).

If there are no ‘timescale directives in the design, uses the value specified for the Resolution variable in *modelsim.ini*.

For SystemC designs, refer to “[Simulation of SystemC Designs](#)” in the User’s Manual for more information.

For mixed-language designs with SystemC, refer to “[Simulator Resolution Limit](#),” in the User’s Manual for more information.

Tip

 After you have started a simulation, you can view the current simulator resolution by entering [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.

If you specify the location of a .tab file, you do not need to use the -no_autoacc argument to prevent 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 argument 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.

- **-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 you omit this argument, the window title is the current Questa SIM version. 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 (refer to [ToggleCountLimit](#) in the User's Manual). Provides default limit values for any design units not compiled with either [vlog](#) -togglecountlimit or [vcom](#) –togglecountlimit.

After reaching the limit, the simulator ignores further activity on the node 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.

Note

 Any design units compiled with vlog -togglecountlimit or vcom -togglecountlimit use those values during simulation unless you use the [toggle add](#) --countlimit command to override the values.

- **-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 (refer to [ToggleWidthLimit](#) in the User's Manual). 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 the actions of the foreign interface side.

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, you must surround the text in quotes. You can use a *modelsim.ini* file variable to set the command for multiple simulations. Valid for use with both VHDL and Verilog designs. If you have VHDL assertions that use “failure” to stop the simulation, you can use this command to keep them from being the worst severity in the simulation. For further information on the application of this argument, see the *Verification Management User's Manual*.

- **-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. Refer to [UndefSyms](#) in the User's Manual.

{<args>}

You must specify at least one argument.

on — (default) Enables automatic generation of stub definitions for undefined symbols, and permits loading of the shared libraries despite the undefined symbols.

off — 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 argument 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.

reseed — Disables behavior of UVM simulation, where if you reseed the simulation, the random sequences generated by UVM will change.

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.

You can specify arguments as multiple instances of -uvmcontrol. Specify multiple arguments 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 final argument overrides earlier arguments and a warning is issued.

You can also control UVM-Aware debugging with the UVMControl modelsim.ini variable. Refer to [UVMControl](#) in the User's Manual.

- **+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.

Note

 You must have an existing VCD file to use this command. To create a VCD file, you must first execute the +dumports+nocollapse or +dumports+collapse options in a Questa SIM simulation, then execute the [vcd dumports](#) command. Refer to “[Using Extended VCD as Stimulus](#)” in the User’s Manual for more information.

- **-verboseprofile**
(optional) Enables enhanced kernel routines that simulate with additional data to improve performance profiler data collection.
- **-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. Enables you to use vsim to view the results from an open simulation (*vsim.wlf*) or an earlier saved simulation. You can open the Structure, Objects, Wave, and List windows to look at the results stored in the WLF file (other Questa SIM windows do 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. Allows wildcard characters.
 - <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> can be:

<class> <depth> — One of the following:

{directcolor | grayscale | greyscale | pseudocolor | staticcolor | staticgray | staticgrey | truecolor}

followed by:

<depth> — Specifies the number of bits per pixel 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 the number of bits per pixel 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 “[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 “[Retain Delay Behavior](#)” in the User’s Manual for more information.

- -voptargs="*<args>*"

(optional) Specifies the arguments for vsim to pass to vopt when running vopt automatically. Specify multiple arguments as a space-separated list.

The primary purpose of this argument is to pass +acc arguments to vopt; you cannot use the vopt -o <name> option.

- -vopt_verbose

(optional) Displays vopt messages in the Transcript window. By default, these messages are not displayed or saved when you run vopt using vsim.

- -vpicompatcb

Reverts to pre-10.5 functionality for how PLI/VPI applications are involved in the simulation. Specifically, when multiple PLI/VPI applications are involved, different applications may register the same kind of callbacks, such as an end of compile callback. The ordering of callbacks between PLI and VPI can become nondeterministic and can be affected by the addition/removal of some other irrelevant PLI/VPI application. Default functionality removes this aspect of nondeterminism.

- -vv

Prints to stdout the C/C++ compile and link subprocess command line 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 [warning](#) and ["Message Severity Level"](#) in the User's Manual 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*. You can also specify this option with the WLFFilename variable in the *modelsim.ini* file. Refer to [WLFFilename](#) in the User's Manual.
 - **-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 can be beneficial in slow network environments. You can also specify this option with the WLFCacheSize variable in the *modelsim.ini* file. Refer to [WLFCacheSize](#) in the User's Manual.
 - **-wlfcollapsedelta**
(default) Instructs Questa SIM to record values in the WLF file only at the end of each simulator delta step, and ignore any sub-delta values. Can reduce WLF file size. You can also specify this option with the WLFCollapseMode variable in the *modelsim.ini* file. Refer to [WLFCollapseMode](#) in the User's Manual.
 - **-wlfcollapsetime**
(optional) Instructs Questa SIM to record values in the WLF file only at the end of each simulator time step, and ignore delta or sub-delta values. Can reduce WLF file size. You can also specify this option 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. Can result in relatively larger WLF files. You can also specify this option with the WLFCollapseMode variable in the *modelsim.ini* file.
 - **-wlfcompress**
(default) Creates compressed WLF files. Use **-nowlfcompress** to turn off compression. You can also specify this option with the WLFCompress variable in the *modelsim.ini* file. Refer to [WLFCompress](#) in the User's Manual.
 - **-nowlfcompress**
(optional) Causes vsim to create uncompressed WLF files. The default is to compress WLF files to reduce file size. This can slow simulation speed by one to two percent. Disabling compression speeds up simulation, and can help if you are experiencing problems with faulty data in the compressed WLF file. You can also specify this option with the WLFCompress variable in the *modelsim.ini* file.

- **-wlfdelteonquit**
(optional) Deletes the current simulation WLF file (*vsim.wlf*) automatically when the simulator exits. You can also specify this option with the **WLFDeleteOnQuit** variable in the *modelsim.ini* file. Refer to [WLFDeleteOnQuit](#) in the User's Manual.
- **-nowlfdelteonquit**
(default) Preserves the current simulation WLF file (*vsim.wlf*) when the simulator exits. You can also specify this option with the **WLFDeleteOnQuit** variable in the *modelsim.ini* file. Refer to [WLFDeleteOnQuit](#) in the User's Manual.
- **-wlfllock**
(optional) Locks a WLF file. An invocation of Questa SIM will not overwrite a WLF file that is being written by a different invocation.
- **-nowlfllock**
(optional) Disables WLF file locking. This prevents vsim from checking whether a WLF file is locked prior to opening it, and prevents 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. You can also specify this option with the **WLFOptimize** variable in the *modelsim.ini* file. Refer to [“Limiting the WLF File Size”](#) in the User's Manual for more information.
- **-nowlfopt**
(optional) Disables optimization of waveform display in the Wave window. You can also specify this option 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 enables you to set one size for the WLF reader cache used during simulation, and a different size for the caches used during post-simulation debug. WLF reader caching caches blocks of the WLF file to reduce redundant file I/O. If you do not specify the **-wlfsimcachesize** argument or the **WLFSimCacheSize** *modelsim.ini* variable, the **-wlfcachesize** argument or the **WLFSimCacheSize** *modelsim.ini* variable settings are used. Refer to [WLFSimCacheSize](#) and [WLFCacheSize](#) in the User's Manual.
- **-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.

You can also specify this option with the WLFSIZELIMIT variable in the *modelsim.ini* file. Refer to [WLFSIZELIMIT](#) and “[Limiting the WLF File Size](#)” in the User’s Manual.

- **-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; optionally, you can specify the resolution by placing 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 two limits takes effect.

You can also specify this option with the WLFTIMELIMIT variable in the *modelsim.ini* file. Refer to [WLFTIMELIMIT](#) in the User’s Manual.

The -wlfslim and -wlftlim arguments are intended to limit WLF file sizes for long or heavily logged simulations. If you use small values for these arguments, they may be overridden by the internal granularity limits of the WLF file format. Refer to “[Limiting the WLF File Size](#)” in the User’s Manual.

VHDL Arguments

- **-absentisempty**

(optional) Specifies to treat any VHDL files as empty that, when opened for read, target non-existent files. Keeps Questa SIM from 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. Also 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. Refer to [AccessObjDebug](#) in the User's Manual.

- **-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](#)” in the User’s Manual 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). Refer to [ToggleNoIntegers](#) in the User’s Manual.

- **-noverboseprofile**

(optional) Disables the routines when used with -autoprofile argument.

-
- **-novhdlvariablelogging**
(optional) Turns off the ability to log recursively or add process variables to the Wave or List windows. Refer to **-vhdlvariable** logging. Refer also to [VhdlVariableLogging](#) in the User's Manual.
 - **-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 can be changed on a per-signal basis. The default value of <int> is 100 values.
 - **+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 can 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**.
 - **-unattemptedimmed**
(optional) Specifies to consider any unexecuted immediate assertions in the coverage calculations for Total Coverage that are displayed in the GUI or in the coverage report. By default, the displayed calculations do not include unexecuted immediate assertions. You can

set this functionality for all simulations through the `UnattemptedImmediateAssertions` *modelsim.ini* file variable. Refer to [UnattemptedImmediateAssertions](#) in the User's Manual.

- `-vcd=[+<option>[+<option>+...]]`

Specifies VCD logging-related option settings. Options are:

`multid` — Enable logging for multi-dimensional arrays.

`maxmultid=<max-limit>` — Override limit on size of multi-dimensional arrays that are logged.

- `-vcdrread <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](#)” in the User's Manual for more details.

- `-vhdlmergedupackage`

(optional) Turns off sharing of one package between two PDUs. Each PDU will have a separate copy of the package. You can also specify this option with the `VhdlSeparatePduPackage` variable in the *modelsim.ini* file. Refer to [VhdlSeparatePduPackage](#) in the User's Manual.

- `-vhdlseparatepdupackage`

(optional, default) Turns on sharing of packages between two or more PDUs.

- `-vhdlvariablelogging`

(optional) Allows 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 argument).

For example, if you specify this `vsim` argument, `log -r /*` will log process variables, as long as `vopt` is specified with `+acc=vand` and the variables are not filtered out by the `WildcardFilter` (by using the "Variable" entry).

You can disable this argument with `-novhdlvariablelogging`. Refer to `-vhdlvariable logging`. Refer also to [VhdlVariableLogging](#) in the User's Manual.

Note

 Logging process variables decreases simulation performance. The recommended procedure is to not log or add process variables to the Wave and List windows. However, if debugging does require logging them, 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 canceled when a new output transition is scheduled. The result is that an output can 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 canceled 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 canceled 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 [ClassDebug](#) in the User's Manual. Refer also to the [classinfo find](#) command.
- **-cvgbintstamp**
(optional) Records the simulation time step (timestamp) at which a covergroup bin was first covered during a simulation run. When you use this argument in simulations, resulting coverage reports will include two additional columns of data: one for timestamp value, and another for the test name which covered the bin. Refer to "[Timestamps and UCDB Modification or Merging](#)" in the User's Manual for usage details.
- **-cvgcollapseembeddedinstances**
Collapses embedded covergroup instances of the same type, when:
 - a. option.per_instance is set to 0
 - b. 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. Refer to [MaxReportRhsSVCrossProducts](#) in the User's Manual.

- **-cvgmergeinstances**
(optional) Sets any *type_option.merge_instances* control to 1 if there is no user-provided default assignment in the source. Overrides the **SVCovergroupMergeInstancesDefault** variable in the *modelsim.ini* file. This option takes effect only on covergroups that do not have any explicit assignment to *type_option.merge_instances* in user code. Refer to [SVCovergroupMergeInstancesDefault](#) and “[IEEE Std 1800-2009 Option Behavior](#)” in the User’s Manual.
- **-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, refer to “[IEEE Std 1800-2009 Option Behavior](#)” in the User’s Manual.
- **-cvgprecollect <ucdb_file>**
(optional) Specifies the UCDB file to use for optimization control in the current simulation. You can use this argument multiple times on the same command line. Refer to [Controlling Functional Coverage Collection](#) in the User’s Manual.
- **-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 <1 | 0>**
(optional) Turn on/off the coverage data collection of any zero-weight coverage items. When turned off, zero-weight coverage items are not displayed in any coverage report and do not 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 GNU compiler supported and distributed by Mentor Graphics to use for the DPI exportwrapper compilation.

<[gcc | g++] version> — The version number of the GNU compiler to use. For example:

`vsim -dpicppinstall 4.5.0`

This overrides the DpiCppInstall variable in the *modelsim.ini* file.

When -cppinstall or -cpppath arguments are also present, the order of precedence in determining the compiler path is the following:

- The -dpicppinstall argument is used with vsim
- The -dpicpppath argument is used with vsim
- The -cppinstall argument is used with vsim
- The -cpppath argument is used with vsim
- **-dpicpppath <pathname>**
(optional) Specifies the explicit path to a gcc compiler to 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. Refer to [DpiCppPath](#) in the User's Manual.
- When -cppinstall or -cpppath arguments are also present, the order of precedence in determining the compiler path is the following:
 - The -dpicppinstall argument is used with vsim
 - The -dpicpppath argument is used with vsim
 - The -cppinstall argument is used with vsim
 - The -cpppath argument is used with vsim
- **-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 can then be included in C source code for DPI import functions. Simulation quits after generation of the header file. Refer to “[DPI Use Flow](#)” in the User's Manual for additional information.
- **-dpilib <libname>**
(optional) Specifies the design library name that contains DPI exports and automatically compiled object files. If you do not set the -dpilib argument, vsim loads export symbols from all libraries accessible using vsim arguments -L, -Lf, and -lib. Multiple occurrences of -dpilib are supported.
- **-dipioutoftheblue 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.

Refer also to the related [DpiOutOfTheBlue](#) *modelsim.ini* file variable in the User's Manual.

- `-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. When the threshold is reached, 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.

Refer to the related [GCThreshold](#) and [GCThresholdClassDebug](#) *modelsim.ini* file variables.

- `-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](#)” in the User's Manual 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. Refer to [LibrarySearchPath](#) in the User's Manual.
- `+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` argument without specifying a modifier (`+{0 | 1 | X | Z}`).

If you do not specify `+initmem` 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 argument when a modifier is not specified (+{0 | 1 | X | Z}).

If you do not specify +initreg 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, 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 can use the delay value based upon the flag you specify with vopt.

- **+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 can use the delay value based upon the flag you specify with vopt.

- **-noappendclose**

(optional) Simulator does 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. Refer to [AppendClose](#) in the User's Manual.

- **+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 the opposite of Verilog-XL, where negative timing check limits are disabled by default, and are enabled with the +neg_tchk argument.

- **+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 with the [tcheck_set](#) command.

- **+no_path_edge**

(optional) Causes Questa SIM to ignore the input edge specified in a path delay. The result 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` arguments. A path pulse error results in a warning message, and the pulse is propagated as an X. The `+no_pulse_msg` argument disables the warning message, but the X is still propagated.

- `-no_risefall_delaynets`

(optional) Disables the default rise/fall delay net delay negative timing check algorithm. This argument returns Questa SIM to older behavior, where violation regions must overlap in order to find a delay net solution. Beginning with Release 6.0, this argument is unnecessary, because 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 do not 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 with the [tcheck_set](#) command.

- `-nocvg`

(optional) Disables covergroup object construction. Also removes the ability to run builtin covergroup methods during simulation. If you set this argument, simulation will fail if there is any hierarchical access to covergroup option variables. This runtime argument is intended to assist in determining if simulation performance is being impacted by covergroup explosion; however, in so determining, it fully disables the SystemVerilog functional coverage feature. Therefore, this argument 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. Overrides the `SVCovergroupMergeInstancesDefault` variable in the `modelsim.ini` file when the variable is set to 1. (Refer to [SVCovergroupMergeInstancesDefault](#) in the User's Manual.) This argument takes effect only 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. Does not generate the `exportwrapper.so` shared object file.

For a description of when you should use this argument, refer to “[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”. You can use “vlog -noexcludeternary <design_unit>” to disable this automatic exclusion for a specific design unit only.
- **-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 repeats until a solution is found. A warning message is issued for each negative limit set to zero.

- **+ntcnotchks**

(optional) Instructs vsim not to simulate timing checks, but to 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 GenerateFormat value controls the previous style. The default behavior is to use the current style names. This argument duplicates the function of the OldVhdlForGenNames variable in modelsim.ini, and overrides the setting of that variable if it specifies the current style. Refer to [GenerateFormat](#), “[Naming Behavior of VHDL For Generate Blocks](#),” and [OldVhdlForGenNames](#) in the User’s Manual.

- **-onElabError [<value>]**

Directs the simulator to take different actions based on the given *value*. Possible values are:

exit — The simulator exits after receiving an elaboration error.

stop — The simulator stops and returns control to the user. If a *do* file script is executing at the time of the error, the script will pause.

resume — The simulator returns control to the running *do* script. If the -do argument was given on the command line and has not yet started executing, vsim will run the *do* script. If no *do* script is given, control returns to the user.

The default behavior when no -onElabError <value> is specified, is to “exit” when running in -c mode, and to “stop” when running in -i or -gui mode.

You can also specify this argument using lowercase: -onelaberror.

- **-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 you to confirmation 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, and prompts you in GUI mode. Edit the OnFinish variable in the *modelsim.ini* file to set the default operation of \$finish. Refer to [OnFinish](#) in the User’s Manual.

- **-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) Enables vsim to elaborate designs that have virtual interface declarations, but do not have actual interface instances that are compatible with the declarations. Such virtual interface declarations are considered "unmatched" since 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" in the User's Manual.

- **-pli "<object list>"**

(optional) Loads a comma- or space-separated list of PLI shared objects. The list must be enclosed in quotes 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](#) in the User's Manual.) You can use environment variables as part of the path.

Optionally, a shared object file entry can contain an initialization function specifier:

<filename>:<init_function>

Use a colon (:) to separate the shared object filename from the initialization function name. Windows systems interpret a colon in the first 3 characters of the *<filename>* as a volume designator. In UNIX, a colon is not reserved and can be specified in the *<filename>*, but a colon is not allowed in the *<init_function>*. For this reason, the rightmost ':' is interpreted to be the separator.

To allow any number of ':' characters in the *<filename>*, a terminating ':' must be added to designate a separator to the end of the argument.

For example, a UNIX directory named "my:dir" containing a shared library named "mypli.sl" is specified as:

```
-pli my:dir/mypli.sl
```

Or the same shared library with an initialization function named "myinit" is specified in UNIX as:

```
-pli my:dir/mypli.sl:myinit
```

- **-plicompatdefault [latest | 2009 | 2005 | 2001]**

(optional) Specifies the VPI object model behavior within vsim. This argument 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 argument, or if you specify the argument without an option.

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 option, 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 overrides the PliCompatDefault variable. Refer to [PliCompatDefault](#) in the User's Manual.

Note that there are cases where the 2005 VPI object model is incompatible with the 2001 model, which is inherent in the specifications.

Refer to "[Verilog Interfaces to C](#)" in the User's Manual for more information.

- +<plusarg>

(optional) Makes the argument following the plus sign (+) accessible to 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 you do not specify the rejection limit, 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. The "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 CELLCODE 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. Refer to [SolveBeforeErrorSeverity](#) in the User's Manual.

Valid values:

- 0 — No error
- 1 — Warning
- 2 — Error
- 3 — Failure (default)
- 4 — Fatal

- **-solveengine {auto | bdd | act}**
(optional) Selects a solver “engine” to use for constrained random generation. Overrides the setting for the SolveEngine variable in the *modelsim.ini* file. (Refer to [SolveEngine](#) in the User's Manual.) 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 can be 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. When a randomize() failure is detected during simulation, Questa SIM displays 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 (1) is enabled.

- **-solvefailsSeverity=<value1>[<value2>]**

(optional) Defines message severity when randomize() and randomize(null) failures are detected. When you specify a single argument, the severity applies to both randomize() and randomize(null). When you specify two arguments (no space between them), the first applies to randomize() and the second applies to randomize(null).

- 0 — (default) No error
- 1 — Warning
- 2 — Error
- 3 — Failure
- 4 — Fatal

- **-solvefailtestcase[=<filename>]**

(optional) Generates a simplified testcase when a randomize() failure is encountered, and outputs the testcase to the specified file. If you do not specify a filename, the testcase will be written to the Transcript. Testcases for 'no-solution' failures will be produced only if you have enabled '-solvefaildebug'.

- **-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) includes 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.) You can use this argument to get the same random sequences during simulation as in 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, for Relase 6.2b you can specify "-solverev 6.2" or "-solverev 6.2a", but you cannot specify "-solverev 6.1f". Refer also to the *modelsim.ini* "[SolveRev](#)" variable in the User's Manual.
- **-solvetimeout <val>**
(optional) Specifies the solver timeout threshold (in seconds). Used 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. Refer also to the *modelsim.ini* "[SolveTimeout](#)" variable in the User's Manual.
- **-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](#)" in the User's Manual for additional information.
- **-sv_liblist <filename>**
(optional) Specifies the name of a bootstrap file containing the names of DPI shared objects (libraries) to load. Refer to "[DPI File Loading](#)" in the User's Manual for format information.
- **-sv_reseed {random | <integer>}**
(Valid only when specified with -load_elab or -restore). Alters the reseed behavior of the -sv_seed argument, as follows:
 - Specified with -load_elab — Behavior is identical to -sv_seed.
 - Specified with -restore — Behavior is to issue the sv_reseed command with the specified argument immediately after the simulation is restored.
 - Specified as part of a UVM-Aware debug flow (requires *questa_uvm_pkg* 1.2.3), the random sequences generated by UVM will change. Can be disabled with vsim -uvmcontrol=reseed.
- **-sv_root <dirname>**
(optional) Specifies the directory name to use 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.

A valid seed value is any non-negative 32-bit integer. Any invalid seed values trigger a warning message and are ignored. Edit the `Sv_Seed` variable in the `modelsim.ini` file to set a permanent default.

Refer to “[Seeding the Random Number Generator \(RNG\)](#)” in the User’s Manual for more information.

Note

 The `Sv_Seed` variable in the `modelsim.ini` file is read-only, and you cannot change it with the `setenv` command (that is, it reflects the current value specified by `restart -sv_seed`).

- `-svext=[+|-]<extension>[,[+|-]<extension>]...`

(optional) Enables or disables non-LRM compliant SystemVerilog language extensions with a comma-separated list of arguments. Any settings to this argument override your settings of the `SvExtensions` `modelsim.ini` variable. Refer to [SvExtensions](#) in the User’s Manual.

- + — activates the *extension*.
- — deactivates the *extension*.

If you do not specify either a “+” or “-”, the command assumes you are activating the specified *extension*.

`<extension>` —

`aptviinexpr` — Allows the parameter reference through virtual interface in part-select expression.

`cfce` — Generates an error message if `$cast` is used as a function and the casting operation fails.

`dfsp` — Sets the default format specifier to `%p` if you do not provide one for unpacked arrays in display-related tasks.

`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 are appended to the return string.

`extscan` — Enables support for values greater than 32 bits in string methods, such as `atohex`, `atoi`, `atobin`, and `atooct`. By default, these methods return a 32-bit value irrespective of the variable type. This extension returns the value indicated by the variable type. For example, given the following:

```
longint d;
string s = "12341231234abcd";
d = s.atohex();
```

by default, `d` will be `64'h000000001234abcd`, where if you specify `extscan`, `d` will be `64'h12341231234abcd`

`fmtcap` — Prints capital hex digits with `%X/%H` in display calls.

iddp — Specifies to ignore the DPI disable protocol check. 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.*

jnds — Schedules the fork/join_none processes at the end of the active region.

lfmt — Zero-pad data if '0' prefixes width in format specifier (for example, "%04h").

pbi — Specifies pointer-based comparison of class handles.

[**-**]realrand — (Questa Simulator Products Only) Enables randomization of ‘real’ variables and constraints, with the exception of multiply and divide with ‘real’ operands in constant expressions. This extension is enabled by default. Use “-svext=realrand” to disable this extension at simulation time. Randomization of ‘real’ variables or constraints requires a SystemVerilog Real Number Modeling (svrnm) license. If the license check fails, ‘real’ number support is disabled.

thrdrngshfl —Determines which random number generator the simulator uses for the array manipulation method (array.shuffle). Setting this to +thrdrngshfl uses a RNG of the active thread, while the default, -thrdrngshfl, uses a separate internal RNG.

- **-svrandext=[+|-]<extension>[,[+|-]<extension>]...**

(optional) Enables or disables non-LRM compliant SystemVerilog constrained random language extensions, with a comma-separated list of arguments. Any settings specified by this argument override your *SvRandExtensions 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> —

arraymode — Enables a non-LRM compliant form of rand_mode for random unpacked arrays (fixed, dynamic, queues, associative). When you enable this extension (along with the vlog or vopt -svext=arraymode extension), every random unpacked array keeps track of its rand_mode value independently of the rand_mode values of its elements. This extension is disabled by default.

deepcheck — Enables calls to class::randomize(null) to consider constraints from contained ‘rand’ class handles.

funcback — Allows functions to be called multiple times, in constraints with complex interactions between the function input(s) and function output (ACT solver only). By default (non-funcback behavior), the solver evaluates random variables related to the input(s) of a function call before random variables that depend on the output of the function call; the function call executes at most once. If a constraint contradiction occurs while evaluating the random variables that depend on the output of the function call, randomize() fails. If you enable funcback, the solver evaluates random variables related to the input(s) and output of a function call simultaneously; the function call can execute multiple times if the output of the function call does not

satisfy the existing constraints. If no solution is found after executing the function 100 times, randomize() fails

nodist — Interprets 'dist' constraint as an 'inside' constraint (ACT solver only).

noorder — Ignores solve/before ordering constraints (ACT solver only).

promotedist — Promotes the priority of a 'dist' constraint if its target has no solve/before constraint.

randindex — (enabled by default) Allows random variables in index expressions for both packed and unpacked arrays within constraints.

randstruct — Treats all fields of an unpacked struct as 'rand', regardless of whether an explicit 'rand' attribute is specified for the field or not.

skew — Skews randomize results (ACT solver only).

strictstab — Enables “strict” random stability, which guarantees that two instances of the same class, having the same randstate, will generate the same randomize() results, regardless of the order in which the randomize() calls are made and independent of any randomize() calls that are made before them. Enabling this option can negatively impact randomize() performance for some scenarios.

- **-togglemaxrealvalues <int>**

(optional) Specifies the maximum number of SystemVerilog real values to record for toggle coverage of a given signal. You can change this limit variable on a per-signal basis. You can modify the default of 100 values by editing the ToggleMaxRealValues *modelsim.ini* variable. Refer to [ToggleMaxRealValues](#) in the User’s Manual.

- **-togglepackedasvec**

(optional) Specifies to treat SystemVerilog packed structures and multi-d arrays as flattened vectors for toggle coverage. Overrides the TogglePackedAsVec *modelsim.ini* variable default setting of off (0). Refer to [TogglePackedAsVec](#) in the User’s Manual.

- **-toggleportsonly**

(optional) Enables only ports for inclusion in toggle coverage numbers; excludes internal signals from coverage metrics. If you want to see coverage of all ports in the design, use the “vopt +acc=p” command — but note that this turns off inlining, and could result in a significant performance penalty. You can selectively enable toggle coverage on specific ports with the “vopt +acc=p+<selection>” command. Overrides any global value set by the TogglePortsOnly *modelsim.ini* variable. Refer to [TogglePortsOnly](#) in the User’s Manual.

- **-togglevlogenumbits**

(optional) Specifies to treat SystemVerilog enum types as reg-vectors for toggle coverage. Overrides the default setting of the ToggleVlogEnumBits variable (off(0)) in the *modelsim.ini*. Refer to [ToggleVlogEnumBits](#) in the User’s Manual.

- **-togglevlogreal**
(optional) Includes Verilog real value types in toggle coverage. Overrides the default setting of the ToggleVlogReal variable (off(0)) in the *modelsim.ini*. Refer to [ToggleVlogReal](#) in the User's Manual.
- **-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 argument works for both Verilog and VITAL cells, though the destination of the interconnect must be a Verilog cell. The source can be VITAL or Verilog. This argument 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 argument 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 can use the delay value based upon the flag specified with **vopt**.
- **-unattemptedimmed**
(optional) Includes any unexecuted immediate assertions in the coverage calculations for Total Coverage displayed in the GUI or in the coverage report. By default, the displayed calculations do not include unexecuted immediate assertions. You can set this functionality for all simulations with the UnattemptedImmediateAssertions *modelsim.ini* file variable. Refer to [UnattemptedImmediateAssertions](#) in the User's Manual.

-
- **-v2k_int_delays**

(optional) Makes interconnect delays 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. You can use this in tandem with the `+multisource_int_delays` argument.

Refer to [sdfcom](#) for SDF compilation information.

- **-work <pathname>**

(optional) When using a 2-step flow, overrides the library in which vsim writes the optimized design generated by the internally invoked `vopt` command.

- **-wrealdefaultzero**

(optional) For nets declared as `wreal`, sets the default value for an undriven `wreal` net to zero (0).

- **-wreal_resolution <val>**

(optional) For Verilog nets declared as `wreal`, selects the resolution function that Questa SIM uses to compute the effective value for multiple drivers on `wreal` nets. Conflicting real values on a `wreal` net are passed through a resolution function, which you specify with one of the following enum values:

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

The resultant value from the computation then appears on the net.

SystemC Arguments

- **-sc22**

(optional) Enables the SystemC-2.2 version (IEEE 1666-2005 standard) for compiling and linking. You can also enable SystemC-2.2 with the [Sc22Mode](#) `modelsim.ini` variable.

- **-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. This scope name renders the design incompatible with OSCI. The `-noscmainscopename` argument strips out the `sc_main` scope name from the output of the `name()` function.

- **-scchkpntrestore**

Enables support for checkpoint and restore commands for a design that contains restricted types of SystemC module (noted below). By default, if a design contains a SystemC module, checkpoint and restore are disabled for the whole design. However, this argument allows checkpoint and restore on the whole design that contains the following types of SystemC modules.

- SystemC module is used as dummy wrapper.
- SystemC region of the design does not contain any active constructs, such as signal, variable, events, or processes.

Note

 If you use this argument with active SystemC regions, Questa SIM displays one or more warnings that this not expected usage.

- **-scdpidebug**

(optional) Enables DPI debug single-stepping across SystemC-SystemVerilog call boundaries for SystemVerilog breakpoints placed inside an export function call initiated from an SC_METHOD.

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 argument is necessary only when the shared library is compiled in a design library other than the logical work directory (using 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 multiplier followed by the size unit, which can be either Kb(kilobyte), Mb(megabyte) or Gb(gigabyte). 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 ScStackSize *modelsim.ini* file variable. Refer to [ScStackSize](#) in the User's Manual.

-
- `-sc_arg <string> ...`

(optional) Specifies a string representing a startup argument, which is subsequently accessible from within SystemC using the `sc_argc()` and `sc_argv()` functions (refer to “[Accessing Command-Line Arguments in SystemC-22](#)” in the User’s Manual).

If you specify multiple SystemC startup arguments, each must have a separate `-sc_arg` argument. SystemC startup arguments are returned by `sc_argv()` in the order in which they appear on the command line. White space within the `<string>` is not treated specially, and the entire string (including white space) is accessible as a single string among the strings returned by `sc_argv()`.

Object Arguments

The object arguments can be a [`<library_name>.<design_unit>`], an `.mpf` file, a `.wlf` file, or a text file. You can specify multiple design units for Verilog modules and mixed VHDL/Verilog configurations.

- `<library_name>.<design_unit>`

(optional) Specifies a library and associated design unit; you can make multiple library/design unit specifications. If you do not specify a library, the work library is used. You cannot use the wildcard character (*) for this argument. You can use environment variables. `<design_unit>` may be one of the following:

<code><c</code> <code>onf</code> <code>igu</code> <code>rati</code> <code>on</code> <code>></code>	Specifies the VHDL configuration to simulate.
<code><m</code> <code>od</code> <code>ule</code> <code>></code>	(optional) Specifies the name of one or more top-level Verilog modules to be simulated.
<code>...</code>	
<code><e</code> <code>ntit</code> <code>y></code> <code>[(<</code> <code>arc</code> <code>hit</code> <code>ect</code> <code>ure</code> <code>>)]</code>	(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

```
<o  (optional) Specifies the
pti name(s) of the optimized
mi design(s). See the vopt
zed command.
_d
esi
gn
_n
am
e>
```

1. Most UNIX shells require arguments containing () to be single-quoted to prevent special parsing by the shell. See the examples below.

You can specify multiple optimized top modules 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 unoptimized design units, the unoptimized design units will not be optimized, and will be elaborated as unoptimized top modules. If all of the top modules are unoptimized, they will be optimized during elaboration. Elaboration of multiple top design units for Power Aware and Schematic debugging is supported only for designs in which all design units are unoptimized.

- <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. To change this behavior so that the Wave window contains all signals in the design, set 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 assign 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* argument 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 arguments 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 argument is ignored. The none option of the second -stats argument disables all modelsim.ini settings and then enables the perf option.

```
vsim -stats=time,cmd,msg -stats=none,perf
```

vsim<info>

Returns information about the current vsim executable.

Syntax

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. You can obtain this same information with the -version argument of the **vsim** command.

Arguments

none

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
```

vsouce

Specifies an alternative file to use for the current source file, when the current source file has been moved. The alternative source mapping applies to the current simulation only.

Syntax

```
vsouce [<filename>]
```

Arguments

- <filename>

(optional) Specifies a relative or full pathname. If you omit the filename, the source file for the current design context is displayed.

Examples

```
vsouce design.vhd  
vsouce /old/design.vhd
```

wave

A collection of related commands that manipulate and report on the Wave window.

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>]
```

Description

The following tables summarize the available options for manipulating cursors, for zooming, and for adjusting the wave display view in the Wave window:

Table 3-13. 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.

Table 3-13. Wave Window Commands for Cursor (cont.)

Cursor Commands	Description
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 to show the specified or active cursor 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 3-14. 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 steps over the full range of the simulation, from time 0 to the current time: <ul style="list-style-type: none"> • Delta time steps if Delta Time mode is currently selected (WLFCollapseMode = 1). • Event time steps if Event Time mode is currently selected (WLFCollapseMode = 0).
wave expand cursor	Expands simulation time into steps at the simulation time of the active cursor: <ul style="list-style-type: none"> • Delta time steps if Delta Time mode is currently selected (WLFCollapseMode = 1). • Event time steps if Event Time mode is currently selected (WLFCollapseMode = 0).
wave expand range	Expands simulation time into steps over a time range specified by a start time and an end time: <ul style="list-style-type: none"> • Delta time steps if Delta Time mode is currently selected (WLFCollapseMode = 1). • Event time steps if Event Time mode is currently selected (WLFCollapseMode = 0).
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 3-15. 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 so 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 so 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 3-16. 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.

Arguments

- **-at <percent>**
(optional) Positions the display so 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) Specifies how much to zoom into or out of the wave display. Default value is 2.0.
- **-lock <0 |1>**
(optional) Specifies the lock state of the cursor.
 0 — (default) Unlocked

1 — Locked

- **-name <name>**
(optional) Specifies the name of the cursor.
 <name> — Any string, where the default is "Cursor <n>" and <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) Specifies a value for the designated option. Currently supported options are -name, -time, and -lock. If no option is specified, reports the current value of all options.
- **<start-time> <end-time>**
(optional) Specifies the start-time and end-time for 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) Specifies to use a Wave window other than the currently active Wave window.
By default, commands run in the currently active Wave window.
 <win> — Specifies the name of a Wave window other than the currently 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

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.

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>
```

Description

Refer to “[Generating Stimulus with Waveform Editor](#)” in the User’s Manual for more information.

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.

Command	Description
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 do not specify a seed value, Questa SIM 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.

Arguments

- **-pattern** {clock | constant | random | repeater | counter | none}

(required) Specifies the waveform pattern. Refer to “[Accessing the Create Pattern Wizard](#)” in the User’s Manual 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 during which 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 you specify <unit>, 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 is 000 and the end value is 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 to use 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 you specify <unit>, you must enclose <time> and <unit> within curly braces ({}).
- **-portmode {input | output | inout | internal}**
 (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. You must specify both.

- **-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. Enter multiple values as a space-separated list 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 you specify <unit>, 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 (for example, for a 3-bit signal, the start value is 000 and the end value is 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 \[Questa SIM User's Manual\]](#)

[Accessing the Create Pattern Wizard \[Questa SIM User's Manual\]](#)

wave edit

Modifies waveforms created with the wave create command.

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>
```

Description

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; does not affect waveform duration
wave edit delete	Delete an edge from a waveform; does not 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

Arguments

- **-backward {<time><unit>}**

(required if -forward <time> is not 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. 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 you specify <unit>, 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 change the specified section of the waveform to. 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. The default is 10 time units.

<unit> — (optional) A suffix specifying a unit of time. 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 you specify <unit>, you must enclose <time> and <unit> within curly braces ({}).

- **-end {<time><unit>}**

(required unless specifying paste) Specifies a simulation time denoting the end of the section of waveform on which to perform the editing operation.

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

<unit> — (optional) A suffix specifying a unit of time. 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. 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 you specify **<unit>**, 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 the final argument to **wave edit**.

- **-start {<time><unit>}**

(required) Specifies a simulation time denoting the beginning of the section of waveform on which to perform the editing operation.

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

<unit> — (optional) A suffix specifying a unit of time. 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 you specify **<unit>**, 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. 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 you specify **<unit>**, 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 \[Questa SIM User's Manual\]](#)

wave export

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 from which to export created waves. If you omit this argument, 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 to stop exporting.

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

 <unit> — (optional) A suffix specifying a unit of time. 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 you specify <unit>, 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 must 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 to start exporting.

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

 <unit> — (optional) A suffix specifying a unit of time. 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 you specify <unit>, you must enclose <time> and <unit> within curly braces ({}).

Related Topics

[wave create](#)

[wave import](#)

[Generating Stimulus with Waveform Editor \[Questa SIM User's Manual\]](#)

wave import

Imports an extended VCD file that was created with the wave export command. It cannot read extended VCD files 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 \[Questa SIM User's Manual\]](#)

wave modify

Modifies waveform parameters set by a previous wave create command.

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>
```

Description

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).

Command	Description
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 do not specify a seed value, Questa SIM 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.

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 the type of the signal driver. Applies to signals of type inout or internal.
- **-dutycycle <value>**
(required) 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) Specifies the simulation time when 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. 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 you specify <unit>, you must enclose <time> and <unit> within curly braces ({}).
- **-endvalue <value>**
(optional) Specifies 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 is 000 and end value is 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 to use to modify 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. You must specify both.
- **-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. 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. Enter multiple values as a space-separated list, enclosed in curly braces ({}).
- **-starttime {<time><unit>}**
(optional) The simulation time at which to start the waveform. 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. 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 you specify `<unit>`, 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. For example, for a 3-bit signal, the start value is 000 and the end value is 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 \[Questa SIM User's Manual\]](#)

[Accessing the Create Pattern Wizard \[Questa SIM User's Manual\]](#)

wave sort

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. If not specified, 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

Instructs Questa SIM to perform actions when specified conditions are met.

Syntax

```
when [[-fast] [-id <id#>] [-label <label>] [-repeat] {<when_condition_expression>} {<command>}]
```

Description

Use this command to control Questa SIM activity for one or more specified conditions.

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 with +acc arguments to enable the setting of breakpoints in the design. Refer to “[Preservation of Object Visibility for Debugging](#)” in the User’s Manual.

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, not related to Tcl, which permits only the operators listed in [Table 3-17](#), and operands that can be HDL object names, signame'event, or constants. Questa SIM evaluates the condition every time any object in the condition changes, hence the restrictions.

The following items can affect your use of 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.
- You cannot use virtual signals, functions, regions, types, and so forth, in the when command. You also cannot use simulator state variables other than \$now.
- You can enter when with no arguments to 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 must use the Tcl uivar command to declare the state. 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

You cannot access channels (such as files, pipes, sockets) that were opened outside of the embedded command. 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 use of 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 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
}
```

Generally, any embedded command done using the -fast argument is global to all other commands used with the -fast argument. In the example above, {\$now == 0ns} defines Tcl processes in a way that the -fast commands can use. These processes have the same restrictions that when bodies have, but the advantage is speed, as a proc tends to execute faster than code in the when body.

It is recommended not to use virtual signals and expressions.

Arguments

- **-fast**
(optional) Causes the embedded <command> to execute in 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](#)”). The 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, adding a level of identification. The label can contain special characters. You must use quotation marks (" ") or braces ({ }) to enclose any <label> that 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 in use, Questa SIM returns 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 in use for a breakpoint.

- `-repeat`

(Limited to “when” breakpoint expressions involving “\$now”). Instructs the command to reestablish the breakpoint after it has been triggered, so that it will fire again for the next time period. Without this argument, expressions using \$now trigger only once.

- `{<when_condition_expression>}`

(required if a command is specified) Specifies the conditions to be met in order to execute the specified `<command>`. The condition is evaluated in the simulator kernel and can be an object name, in which case you can omit the curly braces. The command executes when the object changes value. The condition can be an expression with these operators:

Table 3-17. when_condition_expression Operators

Name	Operator
equals	<code>==, =</code>
not equal	<code>!=, /=</code>
greater than	<code>></code>
less than	<code><</code>
greater than or equal	<code>>=</code>
less than or equal	<code><=</code>
AND	<code>&&, AND</code>
OR	<code> , OR</code>

The operands can be object names, signame'event, or constants. Subexpressions in parentheses are permitted. The command executes 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; you cannot compare the value of two signals. For example, `Name = Name` is not possible.

You can use Tcl variables 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, so 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 for vector types, not the overloaded operators in the `std_logic_1164` package. This can 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 Tcl interpreter within the Questa SIM GUI evaluates command(s) for this argument. Any Questa SIM or Tcl command or series of commands are valid with one exception—you cannot use the `run` command with the when command. The command sequence usually contains a `stop` command that sets a flag to break the simulation run after completion of the command sequence. You can use multiple-line commands.

Note

 If you want to stop the simulation with 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 clk_b_path /tb/ps/dprb_0/udprb/ucar_reg/uint_ram/clk_b;  
when -label when1 "$clk_b_path'event and $clk_b_path ='1" {  
    echo "Detected Clk edge at path $clk_b_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 so that the time unit can 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 to:

```
when "0 = 1000" stop
```

where

Displays information about the system environment. Useful for debugging when Questa SIM cannot find the required libraries or support files.

Syntax

where

Description

The command displays two results on consecutive lines:

- current directory

This is the current directory from which Questa SIM was invoked, 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. For designs that were not loaded through a project, this line displays the *modelsim.ini* file in the current directory.

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

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

 Invoke this command 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]
```

Description

Additional information for QuickSim II users

The results of your original QuickHDL/Questa SIM simulation (in your *vsim.wlf* file) can contain signal values that do not directly correspond to *qsim_12state* values. The QuickSim II logfile that *wlf2log* generates can contain state values that are enclosed in single quotes; for example, '0' and '1'. To make this logfile compatible with QuickSim models (that expect *qsim_12state*), use the QuickSim II *\$convert_wdb()* function.

The *\$convert_wdb()* function converts logfiles from VHDL simulations that use *std_logic* and *std_ulogic*, since these data types do not correlate to the 12 simulation states that QuickSim II recognizes. 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).

To convert a *wlf2log*-generated logfile into a compatible QuickSim WDB while the logfile is 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, "fred," is the name of the new WDB to create. The second argument, 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, save the new WDB ("fred") as a WDB or logfile so you can load it again in the future. The new WDB or logfile contains the correct qsim_12state values, eliminating the need to re-use \$convert_wdb().

Arguments

- <wlffile>
(required) Specifies the Questa SIM WLF file to convert.
- -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. You can combine this argument with the -input, -output, or -internal arguments.
- -input
(optional) Lists only the input ports. You can combine this argument with the -output, -inout, or -internal arguments.
- -internal
(optional) Lists only the internal signals. You can combine this argument with the -input, -output, or -inout arguments.
- -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, displays only the top-level signals.
- -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. You can combine this arguemtns with the -input, -inout, or -internal arguments.

-
- **-quiet**
(optional) Disables error message reporting.

wlf2vcd

Translates a Questa SIM WLF file to a standard VCD file. Complex data types that are unsupported in the VCD standard (records, memories, and so on) are not converted.

Note

 Invoke this command 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 to convert.
- -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

A collection of related commands that enable you to get information about, and perform various actions on, saved WLF files.

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]
```

Description

The different wlfman commands perform the following functions on saved WLF files:

- wlfman info — Returns file information, resolution, versions, and so forth about the source WLF file.
- wlfman items — Generates a list of HDL objects (that is, signals) and/or transaction streams from the source WLF file, and outputs it to stdout. When redirected to a file, the output is referred to as an object_list_file, and can be read in by wlfman filter. Following is an example of an object_list_file:

```
/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
```

The object_list_file is a list of objects and/or transaction streams, one per line. Comments start with a '#' and continue to the end of the line. Wildcards are legal in the leaf portion of the name. 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 include:

- 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 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 you must use the correct 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. Outputs a new line of information each time the state is monitored. The state of the WLF file can be monitored at regular intervals, indicating the changes over time. For example:

```
wlfman monitor visim.wlf
File      Sim
State     Time
closed   14000
```

- wlfman profile — Generates a report with an estimate of the percentage of file space 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 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 estimating the file space each transaction uses as a percentage of total file size. You may be able to reduce WLF file size by not logging some transactions or streams.

The estimated 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, the

estimate assumes that every transaction has 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, not the sizes of the individual elements.

- **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). A merge of two WLF files that contain the same transaction streams records the first stream's data, ignores the second stream, and issues a warning 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 wlfman commands are designed to be used in combination. For example, if you run wlfman profile, and identify that an unneeded signal or transaction stream accounts for 50% of the WLF file size, you can then run wlfman filter to remove the object from the WLF file.

Arguments

- **-o <out_wlffile>**
(required) Specifies the name of the output WLF file. The output WLF file contains 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>. Transactions that start prior to the specified time are ignored.
- **-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 has 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> stops 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.
- -i <intervalTime>
 - (optional) Specifies the time delay before the next status update. 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. You can add indexing and optimization information back to the file with 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 is 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. The default is to include all objects.
- **-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 to display only the top <number> signals in terms of file space percentage.
- **-v**
(optional) Produces verbose output listing 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          1224     18 %    /top_vh/ptrans
#      1          1216     18 %    /top_vh/sdata
#      3          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 1000000000 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 \[Questa SIM User's Manual\]](#)

[WLF File Parameter Overview \[Questa SIM User's Manual\]](#)

wlfrecover

Attempts to "repair" WLF files that are incomplete because of a crash, or because 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 \[Questa SIM User's Manual\]](#)

[write cell_report](#)

Writes a report to the Transcript window, or to a file, listing Verilog modules which qualified for and received gate-level cell optimizations.

Note

 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 you do not specify a filename, the [write report](#) command is 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 output file.

Related Topics

[Reports for Gate-Level Optimizations \[Questa SIM User's Manual\]](#)

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.

Syntax

```
write format {<window_type>} [-window <window_name>] <filename>  
write format restart [<option option1 ...>] <filename>
```

Description

The write command creates a file created that is primarily a list of [add list](#), [add wave](#), and [configure](#) commands, but it includes a few other commands (refer to "Output" below).

You can invoke this command with the [do](#) command to recreate the window format on a subsequent simulation run (refer to [restart](#) below).

Arguments

- <window_type>
(required unless specifying restart) Specifies to record the contents of <window_type> in the file designated 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 records all debug windows, all file/line breakpoints, and all signal breakpoints created using the [when](#) command. The ShutdownFile *modelsim.ini* variable, when set to this .do filename, calls the write format restart command upon exit. Refer to [ShutdownFile](#) in the User's Manual.

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 when not all datasets exist, use the -force switch with your do command. For example:

```
VSIM> do format.do -force
```

Note

 Note that if you use the -force switch when some datasets do not exist, you will get 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 which List or Wave window to record the contents of. Use when you have more than one instance of the List or Wave window.

- <filename>

(required) Specifies the name of the output file to write the data to. 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
```

- 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, the *-noupdate* argument adds five signals 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. Only saved Wave format files use these four commands; they are not documented elsewhere.

write list

Records the contents of the List window in a list output file.

Syntax

write list [-events] [-window <wname>] <filename>

Description

The list output file contains simulation data for all HDL objects displayed in the List window: VHDL signals and variables, and Verilog nets and registers.

Arguments

- **-events**
(optional) Specifies to write the output in print-on-change format. The default is tabular format.
- **-window <wname>**
(optional) Specifies an instance of the List window that is not the default. If you do not specify this argument, the command uses the default List window. 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 to write the data to.

Examples

- Save the current data in the 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

Saves the current GUI preference settings to a Tcl preference file. The saved setting 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. Refer to [MODELSIM_TCL](#) in the User's Manual.

You can modify variables by editing the preference file with the Questa SIM [notepad](#):

`notepad <preference file name>`

write report

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.

Syntax

```
write report [-capacity [-l | -s] [-line] [-assertions | -classes | -cvg | -qdbs | -solver | -vmem]] |  
[-l | -s] | [-solver] | [-tcl] | [<filename>]
```

Description

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

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 includes a switch that you can specify, along with **-capacity**, 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 to write the data to. If you omit <filename>, the report is written to the Transcript window.
- **-l**
(optional) Generates 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.
- **-line**
(optional) Generates point of allocation (line) based report. If you do not specify -line, the report is generated based on declaration. Vsim must be run with -capacity=line to print a point-of-allocation (line) based report.
- **-qdas**
(optional) Reports memory usage data for queues, dynamic arrays, associative arrays, and strings (each in its own section in the report). 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) Specified with capacity, -solver reports memory usage data for calls to randomize() and memory usage.
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. You cannot use this argument when generating an output file with the <filename> argument.
- **-vmem**
(optional) When specified with capacity, -vmem reports usage data for Verilog memories.

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  
  
...
```

- Create a Tcl-based report for the current simulation:

write report -tcl

which creates output similar to:

```
{1 {Package Body} <path>/ieee_std_logic_textio {} {<path>/vhdl_src/  
synopsys/std_logic_textio.vhd} notAcell}  
{4 Entity <path>/dataflow/work cache_set only {set.vhd util.vhd}  
notAcell}  
{4 Module <path>/dataflow_verilog/work cache_set fast {set.v}  
notAcell}
```

where the output is of the format:

```
{<occurrences> <type> <library_location> <name> <architecture>  
<source_filename> ...} <cell_info>}
```

write timing

Displays path delays and timing check limits, unadjusted for delay net delays, for the specified instance.

Syntax

```
write timing [-recursive] [-file <filename>] [<instance_name1>...<instance_nameN>]  
[-simvalues]
```

Description

The write timing command reports interconnect delays on a Verilog module instance as either MIPDs (Module Input Port Delays) or MITDs (Module Transport Port Delays). If you specify either the `+multisource_int_delays` or the `+transport_int_delays` argument with the `vsim` command, INTERCONNECT delays are reported as MITDs. Otherwise they are reported as MIPDs.

Following is an example MIPD report:

```
# /top/u1: [mymod:src/5/test.v(18)]  
# MIPD(s):  
#   Port clk_in: (6, 6, 6)
```

Following is an example MITD report:

```
# /top/u1: [mymod:src/5/test.v(18)]  
# MITDs to port clk_in:  
# From port /top/p/y = (6)
```

When you specify the `+multisource_int_delays` argument without `+sdf_verbose` on the `vsim` command line, "write timing" does not report the individual bits of vector source ports of SDF INTERCONNECT delays.

For example, assume the SDF file contains the following two INTERCONNECT statements:

```
(INTERCONNECT p/y[0] n/bus_in[0] (3))  
(INTERCONNECT p/y[1] n/bus_in[1] (4))
```

The corresponding "write timing" output is:

```
# MITDs to port bus_in[0]:  
# From port /tb12/p/y = (3)  
# MITDs to port bus_in[1]:  
#   From port /tb12/p/y = (4)
```

Notice that the report does not include the specific bits of the source port.

If you add "+sdf_verbose" to the vsim command line, the "write timing" output becomes:

```
# MITDs to port bus_in[0]:  
# From port /tb12/p/y[0] = (3)  
# MITDs to port bus_in[1]:  
#   From port /tb12/p/y[1] = (4)
```

The report now includes the specific bits of the source port.

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

Writes the contents of the Transcript window to a file. You can then modify the resulting file to replay the transcribed commands as a DO file (macro).

Note

 You cannot use this command 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 to write the data to. If you omit <filename>, the transcript is written to a file named *transcript*.

Related Topics

[Saving a Transcript File as a DO file \[Questa SIM GUI Reference Manual\]](#)

write tssi

Records the contents of the default or specified List window in a “TSSI format” file.

Note

 TSSI format is sometimes referred to as SEF format. The two terms are interchangeable.

Syntax

`write tssi [-window <wname>] <filename>`

Description

The write tssi command creates a file that contains TSSI simulation data for all HDL objects displayed in the List window that can be converted to TSSI format (VHDL signals and Verilog nets). It also generates a signal definition file.

The List window must be using symbolic radix in order for write tssi to produce useful output.

If <filename> has a file extension (for example, *listfile.lst*), the definition file uses the same filename with the extension *.def* (for example, *listfile.def*). The values appear in the listfile in the same order in which they appear in the List window. If the object is a port, the port type determines the directionality. Otherwise, the object is assumed to be bidirectional (mode INOUT).

Objects that the write tssi command can convert to TSSI format are VHDL enumerations with 255 or fewer elements and Verilog nets. The command converts enumeration values U, X, 0, 1, Z, W, L, H, and - (the enumeration values defined in the IEEE Standard 1164 std_ulegic enumeration) to TSSI values as shown in the table below.

std_ulegic State Characters	TSSI (SEF) State Characters		
	Input	Output	Bidirectional
U	N	X	?
X	N	X	?
0	D	L	0
1	U	H	1
Z	Z	T	F
W	N	X	?
L	D	L	0
H	U	H	1
-	N	X	?

Lowercase values x, z, w, l, and h are converted to the same values as the corresponding capitalized values. Other values are converted to a question mark (?) and cause an error message.

Though the write tssi command was developed for use with std_ulegic, it also converts any signal that uses only the values defined for std_ulegic (including the VHDL standard type bit).

Bidirectional logic values are not converted because only the resolved value is available. You can use the TSSI TDS ASCII In Converter and ASCII Out Converter to resolve the directionality of the signal, and to determine the proper forcing or expected value on the port.

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.
-

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.

write wave

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 to end the record.

 <time> — Specified as a positive integer or decimal number. 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. 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. 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. The units are the current simulation time resolution.
- -portrait
(optional) Use portrait (vertical) orientation. The default is landscape (horizontal).
- -start <time>
(optional) Specifies the start time to be written.

 <time> — Specified as a positive integer or decimal number. 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. The default is 8.5.

- **-window <wname>**

(optional) Specifies an instance of the Wave window that is not the default. Otherwise, uses the default Wave window. 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 ease the job of creating a PostScript waveform output file, you can use the **File > Print Postscript** menu selection in the Wave window.

xml2ucdb

Converts an XML testplan file to a *.ucdb* file. The configuration settings for this utility are read automatically from the *xml2ucdb.ini* file in the <install_dir>/vm_src/ directory.

Syntax

```
xml2ucdb [<options>] <XML_filename> {<ucdb_filename> | -ucdbfilename <file>}
```

Where <options> are:

```
[ -help ]  
[ -autonumber | -noautonumber ] [ -autoweight ]  
[ -cfgfilename <path/ini_file_name> ] [ -createcovitem | -nocreatecovitem ]  
[ -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> ]
```

Description

The settings specified by this command override any settings in the *xml2ucdb.ini* file (refer to “[xml2ucdb.ini Configuration File](#)” in the Verification Management User’s Manual).

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 you use this command in scripts, the returned value can help you determine what actions to take next, depending on the success of the command.

Arguments

- **-autonumber | -noautonumber**
(optional) Enables (-autonumber) or disables (-noautonumber) the automatic generation of testitem numbers from section tags. By default, autonumbering is disabled. Use -autonumber to enable it. After overriding the default off setting by enabling autonumbering in a custom configuration (*xml2ucdb.ini*) file, you can use -noautonumber to subsequently turn it 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, refer to “[Coverage Calculation in Testplans](#)” in the Verification Management User’s Manual.

- **-createcovitem | -nocreatecovitem**

(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 does not 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, refer to “[Counting the Coverage Contribution from Unimplemented Links](#)” in the Verification Management User’s Manual.

By default, this functionality is enabled. Use `-nocreatecovitem` to disable the default behavior.

- **-cfgfilename <path/ini_file_name>**

(optional) Specifies an alternative XML2UCDB configuration file to use (that is, 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, can 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 is treated by QuestaSim as a kind of “reserved” word. These case sensitive, order dependent, reserved words must match those listed in the table of default expected columns/fields shown in the “[Fields in the Verification Plan](#)” section of the Verification Management User’s Manual. 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. You can also use this switch to add additional, user-defined data fields to the testplan. Specify blank fields in XML source with the null character (“-”).

For more information on the datafields in the default `xml2ucdb.ini` file, located in `<install_dir>/vrm_src/`, refer to “[Parameter for Mapping by Column Sequence](#)” in the Verification Management User’s Manual.

- **-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, the testplan mapping file, that contains coverage CLI commands used to link testplan with coverage items. The testplan mapping file contains the coverage tag commands necessary to tag and link the coverage objects to the testplan items. The default is no mapping file.
- **-excludesections {<sec>[.<sec>][-<sec>[.<sec>]]},[<...>]**
(optional) Specifies section(s) of the source plan to exclude from the generated testplan UCDB. You can specify the sections with 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, and their children.

- **-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> must match the string that appears in the tab (“Sheet1”, “Sheet2”, and so on) at the bottom of an Excel spreadsheet.
- **-format {<format>}**
(optional) Specifies the format use 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.

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, and so on), 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. If you use **-format**, you do not need to define all the parameters specific to a particular format within the *xml2ucdb* command itself (-datafields, -datalabels, -datatags, and so on). For more details, refer to “[Verification Plan Contents](#)” in the Verification Management User’s Manual.

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 MVCEExcel (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 values 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. Do not place a space between -G and <varname>, or 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, and replaces the variable *var1* with the value “*val 1*”. For more information, refer to “[Parameterizing Testplans](#)” in the Verification Management User’s Manual.
- **-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 to replace 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 filename. On Windows systems, the path separator is a forward slash (/).
- **-root <str>**
(optional) Specifies the root name to prepend to each section number in non-autonumbered testplans (that is, spreadsheets) of the testplan. Enables you to specify a root testplan in which to “nest” another testplan, thereby creating a hierarchical testplan. If you do not specify -root, 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 you do not specify this switch for a hierarchical (nested) testplan, any existing setting for parent testplan(s) is inherited. If there is no setting 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. Default <num> is 0.
- **-startstoring <num>**
(optional) Specifies the Storing to begin at an item number. Default <num> is 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 you do not set the tagprefix , the value specified with -title is used. Any whitespace contained in the title 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 to use as tag separators for tag arguments accepting multiple tags. Applies only to the taglist parameters (-starttags, -stoptags, excludetags, and so on) specified on the command line.
- **-title <str>**
(optional) Specifies a string to use as the title for the testplan. For a top-level testplan, if you do not set the title, 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, and so on). 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 any values being used for parameters.
- **-varfile <path/varfile_name>**
(optional) Name of a file containing the variables for substitution within plan. Must include the path if the varfile is not in the same directory in which the xml2ucdb command is issued. The <varfile_name> contains a list of <variable>=<value> entries, one per line. Spaces in <variable> are ignored; spaces in <value> are honored. You can override the specifications made by this argument with -G<varname>=<value>. For more information, refer to "[Parameterizing Testplans](#)" in the Verification Management User's Manual.
- **-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 convert. The path can be a full or relative path to the file location. On Windows systems, the path separator must 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 \[Questa Verification Management User's Manual\]](#)

[XML Terms / Concept Review for Plan Import \[Questa Verification Management User's Manual\]](#)

[xml2ucdb.ini Configuration File \[Questa Verification Management User's Manual\]](#)

[Parameterizing Testplans \[Questa Verification Management User's Manual\]](#)

xprop assertlimit

Sets a fail count limit for X-propagated assertions that trigger at every interval. Prevents these X-propagated assertions from slowing down the simulation and making it noisy. Used with simulations that you run with vopt -xprop.

Syntax

`xprop assertlimit <n>`

Arguments

- `<n>`

Specifies an integer as the maximum number of times an xprop assertion is triggered. Default value is 5. To allow an unlimited number of assertions, set `<n>` to -1.

This number overrides any value you set with the `XpropAssertionLimit` *modelsim.ini* variable. Refer to [XpropAssertionLimit](#) in the User's Manual.

Related Topics

[X Propagation in Simulation \[Questa SIM User's Manual\]](#)

[XpropAssertionLimit \[Questa SIM User's Manual\]](#)

xprop disable

Disables the xprop effect, which is a pessimistic, gate-level simulation (GLS) mode of propagating X values through a design. After you disable the xprop effect, the simulation runs with the default RTL X-propagation, which is a more optimistic mode. Used on designs previously optimized using vopt -xprop.

Syntax

`xprop disable`

Arguments

None

Description

Note

 Do not use `xprop enable` and `xprop disable` with assertion enable -on/-off. Use one command family consistently.

Related Topics

[X Propagation in Simulation \[Questa SIM User's Manual\]](#)

xprop enable

Re-enables 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

Description

Note

 Do not use xprop enable and xprop disable with assertion enable -on/-off. Use one command family consistently.

Related Topics

[X Propagation in Simulation \[Questa SIM User's Manual\]](#)

Index

— Symbols —

.main clear command, 72
'delayed, 45
'hasX, 45
'hasX, hasX, 45
+define+, 904
+delay_mode_distributed, 905, 954
+delay_mode_path, 905, 954
+delay_mode_unit, 905, 954
+delay_mode_zero, 905, 954
+inmdir+, 910
+libcell, 914
+ma x dela ys, 916
+maxdelays, 968
+mindelays, 916, 969
+nolibcell, 914, 972
+nowarn, 920, 974
+typdelays, 932, 1004
<\$no page, 995
\$finish behavior, customizing, 1064
\$finish behavior, customizing, 1064

— A —

abort command, 73
absolute time, using @, 35
add _menuitem simulator command, 106
add atv command, 74
add button command, 76
add dataflow command, 77
add list command, 79
add log command, 421
add memory command, 84
add message command, 86
add schematic command, 88
add testbrowser command, 90
add watch command, 91
add wave command, 92
add_cmdhelp command, 100
add_menu command, 102

add_menuacb command, 104
add_separator command, 108
add_submenu command, 109
addTime command, 626
alias command, 111
analog
 signal formatting, 94
annotating interconnect delays,
 v2k_int_delays, 1074
archive load command, 112
archive write command, 113
arrays
 slices, 22
arrays
 indexes, 22
arrays, VHDL, searching for, 40
assertion action command, 114
assertion active command, 117
assertion count command, 119
assertion enable command, 121
assertion fail command, 125
assertion pass command, 128
assertion profile command, 130
assertion report command, 132
assertions
 count, 119
 enable, 121
 enabling, 125, 128
 failure behavior, 125
 pass behavior, 128
 profiling, 130
 reporting, 132
 testing for with onbreak command, 474
 thread viewing, 74, 136
assume directives
 -assume argument, 1014
 -noa ssume argument, 1032
attributes, of signals, using in expressions, 43
atv log command, 136

— B —

batch_mode command, 138
batch-mode simulations
 halting, 1111
bd (breakpoint delete) command, 139
binary radix, mapping to std_logic values, 52
bookmark add wave command, 141
bookmark delete wave command, 143
bookmark goto wave command, 144
bp (breakpoint) command, 146
break
 on signal value, 1104
breakpoints
 conditional, 1104
 continuing simulation after, 566
 deleting, 139
 listing, 146
 setting, 146
 signal breakpoints (when statements), 1104
 time-based
 in when statements, 1111
bus contention checking, 166
 configuring, 168
 disabling, 169
bus float checking
 configuring, 172
 disabling, 173
 enabling, 170
busses
 user-defined, 98
buttons, adding to the Main window toolbar, 76

— C —

case choice, must be locally static, 760
case sensitivity
 VHDL vs. Verilog, 28
Ccallstack
 moving down, 534
 moving up, 487
cd (change directory) command, 158
cdbg command, 159
Cdebugging, 159
change command, 162
change_menu_cmd command, 165
check contention add command, 166

check contention config command, 168
check contention off command, 169
check float add command, 170
check float config command, 172
check float off command, 173
check stable off command, 174
check stable on command, 175
-check_synthesis argument, 749
checkpoint command, 177
class instance garbage collector, 405, 407
class member selection, syntax, 22
class objects, viewing, 183, 185, 187, 190, 193, 195, 197, 199
classinfo command, 183, 185, 187, 190, 193, 195, 197, 199
Code Coverage
 coverage analyze command, 245
 coverage clear command, 260
 coverage exclude command, 272
 coverage goal command, 283
 coverage report command, 298
 coverage save command, 311
 coverage tag command, 315
 coverage testnames command, 322
 coverage unlinked command, 323
 coverage weight command, 327
 merging reports, 788
 toggle coverage
 excluding signals, 633
 vcover report command, 818
Color
 radix, 547
 example, 548
combining signals, busses, 98
comm ands
 compare add, 201
command line args, accessing
 vopt sc_arg command, 998
 vsim sc_arg command, 1076
commands
 .main clear, 72
 abort, 73
 add atv, 74
 add button, 76
 add dataflow, 77

add list, 79
add memory, 84
add message, 86
add schematic, 88
add testbrowser, 90
add watch, 91
add wave, 92
add_cmdhelp, 100
add_menu, 102
add_menucb, 104
add_menuitem, 106
add_separator, 108
add_submenu, 109
alias, 111
archive load, 112
archive write, 113
assertion action, 114
assertion active, 117
assertion count, 119
assertion enable, 121
assertion fail command, 125
assertion pass, 128
assertion profile, 130
assertion report, 132
atv log, 136
batch_mode, 138
bd (breakpoint delete), 139
bookmark add wave, 141
bookmark delete wave, 143
bookmark goto wave, 144
bp (breakpoint), 146
cd (change directory), 158
cdbg, 159
change, 162
change_menu_cmd, 165
check contention config, 168
check contention add, 166
check contention off, 169
check float add, 170
check float config, 172
check float off, 173
check stable off, 174
check stable on, 175
checkpoint, 177
classinfo, 183, 185, 187, 190, 193, 195, 197, 199
compare annotate, 207
compare clock, 209
compare configure, 211
compare continue, 213
compare delete, 214
compare end, 215
compare info, 216
compare list, 218
compare options, 219
compare reload, 224
compare reset, 225
compare run, 226
compare savediffs, 227
compare saverules, 228
compare see, 229
compare start, 226, 231
compare stop, 233
compare update, 234
configure, 235
context, 241
coverage analyze, 245
coverage attribute, 255
coverage clear, 260
coverage create, 263
coverage edit, 266
coverage exclude, 272
coverage goal, 283
coverage loadtestassoc, 287
coverage open, 288
coverage ranktest, 289
coverage report, 298
coverage save, 311
coverage tag, 315
coverage testnames, 322
coverage unlinked, 323
coverage weight, 327
dataset clear, 332
dataset close, 334
dataset config, 335
dataset current, 337
dataset info, 338
dataset list, 339
dataset open, 340

dataset rename, 341
dataset restart, 342
dataset save, 343
dataset snapshot, 344
delete, 347
describe, 348
disablebp, 350
do, 353
down, 355
drivers, 357
dumplog64, 359
echo, 360
edit, 361
enable_menu, 363
enable_menuitem, 364
enablebp, 362
encoding, 365
environment, 366
examine, 368
exit, 376
fcover configue, 377
find, 381
find connections, 386
find drivers, 387
find infiles, 392
find insource, 393
force, 395
fsm list, 402
fsm properties, 403
fsm view, 404
gc configure, 405
gc run, 407
gdb dir, 408
getactivecursortime, 409
getactivemarkertime, 410
help, 411
history, 412
jobspy, 413
layout, 415
lecho, 417
left, 418
log, 421
lshift, 425
lsublist, 426
mem compare, 427
mem display, 428
mem list, 431
mem load, 432
mem save, 436
mem search, 439
next, 465
noforce, 466
nolog, 467
notepad, 469
noview, 470
nowhen, 471
onbreak, 472
onElabError, 474
onerror, 475
onfinish, 477
pa autotestplan, 478
pause, 486
pop, 487
power report, 496
power reset, 499
printenv, 501, 502
process report, 503
profile clear, 504
profile off, 508
profile on, 510
profile open, 512
profile option, 513
profile reload, 515
profile save, 521
profile summary, 525
property list, 530
property wave, 532
push, 534
pwd, 535
questasim, 536
quietly, 537
quit, 538
qverilog, 539
radix, 543
radix define, 546
radix list, 551
radix name, 552
readers, 554
report, 555
restart, 558

restore, 560
resume, 561
right, 562
run, 565
runStatus, 568
sccom, 570
scgenmod, 581
sdfcom, 585
search, 587
searchlog, 590
see, 593
seetime, 594
setenv, 595
shift, 596
show, 597
simstats, 598
simstatslist, 601
stack down, 603
stack frame, 604
stack level, 605
stack tb, 606
stack up, 607
status, 608
stop, 611
suppress, 612
sv_reseed, 614
tb (traceback), 606, 615
tcheck_set, 616
tcheck_status, 621
Time, 626
toggle add, 629
toggle disable, 633
toggle enable, 635
toggle report, 636
toggle reset, 638
tr color, 639
tr order, 642
tr uid, 644
transcribe, 646
transcript, 647
transcript file, 648
transcript path, 650
transcript sizelimit, 651
TreeUpdate, 1129
tssi2mti, 675
typespec, 676
unsetenv, 679
up, 680
uvm call, 683
uvm configtracing, 686
uvm displayobjections, 688
uvm findregisters, 691
uvm findsequences, 694
uvm handle, 696, 698
uvm printconfig, 700
uvm printfactory, 703
uvm printstreams, 706
uvm printtopology, 708
uvm setverbosity, 710
uvm simpAth, 712
uvm traceobjections, 714
uvm uvmpath, 716
variables referenced in, 35
vcd add, 718
vcd checkpoint, 721
vcd comment, 722
vcd dumpports, 723
vcd dumpportsall, 726
vcd dumpportsflush, 727
vcd dumpportslimit, 728
vcd dumpportsoff, 730
vcd dumpportson, 731
vcd file, 732
vcd files, 734
vcd flush, 737
vcd limit, 739
vcd off, 741
vcd on, 742
vcom, 747
vcover attribute, 772
vcover dump, 782
vcover history, 785
vcover merge, 788
vcover parallelmerge, 798
vcover ranktest, 803
vcover remove, 814
vcover report, 818
vcover testnames, 843
vdel, 847
vdir, 849

vencrypt, 852
verror, 859
vgencomp, 861
vhencrypt, 863
view, 866
virtual count, 870
virtual define, 871
virtual delete, 872
virtual describe, 873
virtual expand, 874
virtual function, 875
virtual hide, 879
virtual log, 880
virtual nohide, 882
virtual nolog, 883
virtual region, 885
virtual save, 886
virtual show, 887
virtual signal, 888
vlib, 894
vlog, 898
vmake, 936
vmap, 939
vsim, 1012
vsimVersion, 1079
vsource, 1081
wave, 1082
wave create, 1086
wave edit, 1092
wave export, 1095
wave import, 1097
wave modify, 1098
wave sort, 1103
WaveActivateNextPane, 1129
WaveRestoreCursors, 1129
WaveRestoreZoom, 1129
when, 1104
where, 1112
wlf2log, 1113
wlf2vcf, 1116
wlfman, 1117
wlfrecover, 1124
write cell_report, 1125
write list, 1129
write preferences, 1130
write report, 1131
write timing, 1135
write transcript, 1137
write tssi, 1138
write wave, 1140
xml2ucdb, 1142
xprop assertlimit, 1149
xprop disable, 1150
xprop enable, 1151
commands formatTime, 401
comment characters in VSIM commands, 20
compare add command, 201
compare annotate command, 207
compare clock command, 209
compare configure command, 211
compare continue command, 213
compare delete command, 214
compare end command, 215
compare info command, 216
compare list command, 218
compare options command, 219
compare reload command, 224
compare reset command, 225
compare run command, 226
compare savediffs command, 227
compare saverules command, 228
compare see command, 229
compare start command, 226, 231
compare stop command, 233
compare update command, 234
compatibility, of vendor libraries, 849
Compile
 order of precedence, 572, 573, 903, 905,
 906, 948, 949, 1018, 1058
compiling
 range checking in VHDL, 765
 SystemC, 570, 581
 Verilog, 898
 VHDL, 747
 selected design units (-just eapbc), 757
 standard package (-s), 766, 925
 VHDL-2008
 REAL_VECTOR, 756
compressing files
 checkpoint files, 177

elaboration files, 1017
VCD files, 723, 734

concatenation
 directives, 50
 of signals, 49

conditional breakpoints, 1104

conf igu rations, simul ating, 1012

configurations, simulating, 1012

configure command, 235

constants
 in case statements, 760
 values of, displaying, 348, 368

contention checking, 166

context command, 241

conversion
 radix, 543

coverage
 vcover testnames command, 843

coverage analylze command, 245

coverage attribute command, 255

coverage clear command, 260

coverage create command, 263

coverage edit command, 266

coverage exclude command, 272

coverage goal command, 283

coverage loadtestassoc command, 287

coverage open command, 288

coverage ranktest command, 289

coverage report command, 298

coverage save command, 311

coverage tag command, 315

coverage testnames command, 322

coverage unlinked command, 323

Coverage View mode
 coverage loadtestassoc command, 287
 coverage open command, 288

coverage weight command, 327

customizing
 adding buttons, 76

— D —

dataset clear command, 332

dataset close command, 334

dataset config command, 335

dataset current command, 337

dataset info command, 338

dataset list command, 339

dataset open command, 340

dataset rename command, 341

dataset restart command, 342

dataset save command, 343

dataset snapshot command, 344

datasets
 environment command, specifying with, 366

de sign loading, interr upting, 1012

declarations, hiding i mplicit with explicit, 770

default VOPT behavior, and .ini file, 943

delay
 interconnect, 1031

delete command, 347

deltas
 collapsing in WLF files, 1050

dependencies, checking, 849

dependency errors, 753, 908

describe command, 348

design units
 report of units simulated, 1131

Verilog
 adding to a library, 898

directories
 mapping libraries, 939

disablebp command, 350

dividers
 adding from command line, 93

divTime ccommand, 626

do command, 353

DO file
 executing, 353

DO files, 353
 breakpoints, executing at, 148
 forcing signals, nets, or registers, 395
 parameters
 passing, 353
 relative directories, 353
 shifting parameter values, 596

down command, 355

-dpih eader, vopt, 955

-dpihheader, vlog, 906, 1058

drivers command, 357

dump files, viewing in the simulator, 744

dumplog64 command, 359

— E —

echo command, 360
edges, finding, 418, 562
edit command, 361
enable_menu command, 363
enable_menuitem command, 364
enablebp command, 362
encoding command, 365
encryption
 +protect argument, 924
 -nodebug argument (vcom), 760
 -nodebug argument (vlog), 918
entities, specifying for simulation, 1077
environment command, 366
environment variables
 reading into Verilog code, 904
 specifying UNIX editor, 361
 state of, 502
 using in pathnames, 28
environment, displaying or changing
 pathname, 366
eqTime command, 626
errors
 getting details about messages, 859
 onerror command, 475
 SDF, disabling, 1041
event order
 changing in Verilog, 900, 903, 948
examine command, 368
exit command, 376
exiting the simulator, customizing behavior,
 1064
exiting the simulator, customizing behavior,
 1064
extended identifiers, 28
extended toggle coverage, reporting, 300, 819

— F —

fcover configue command, 377
file compression
 checkpoint files, 177
 elaboration files, 1017
 VCD files, 723, 734
find command, 381

find connections command, 386

find drivers command, 387
find infiles command, 392
find insource command, 393
fixed point radix, 546
floating point radix, 546
force

 remove wire model, 1035

force comma nd, 395

force command, 395

foreign module declaration

 Verilog example, 585

formatTime command, 401, 626

fsm list command, 402

fsm properties command, 403

fsm view command, 404

Functional coverage

 merging databases offline, 788

functional coverage

 configuring directives, 377

— G —

gc configure command, 405
gc run command, 407
gdb dir command, 408
generics
 limitation on assigning composite types,
 1023
generics
 assigning or overriding values with -g and -
 G, 1022
 examining generic values, 368
 limitation on assigning composite types,
 960
getactivecursortime command, 409
getactivemarkertime command, 410
glitches
 disabling generation
 from command line, 1053
global visibility
 PLI/FLI shared objects, 1024
 shared objects, 961
gotolingk questa_sim_user
 DPI File Loading, 1069
gteTime command, 626
gtTime command, 626

GUI_expression_format, 42
syntax, 43

— H —

hazards
-hazards argument to vlog, 961
hazards
-hazards argument to vlog, 910
-hazards argument to vsim, 1059
help command, 411
history
of commands
shortcuts for reuse, 39
history command, 412

— I —

implicit operator, hiding with vcom -explicit, 770
interconnect de lays, 1031
interconnect delays
annotating per Verilog 2001, 1074
internal signals, adding to a VCD file, 719
interruptingdesignloading, 1012
intToTime command, 626

— J —

jobspray command, 413

— K —

keywords
enabling SystemVerilog keywords, 927

— L —

layout command, 415
LD_LIBRARY_PATH, disabling default internal setting of, 1032
lecho command, 417
left command, 418
libraries
dependencies, checking, 849
design libraries, creating, 894
listing contents, 849
refreshing library images, 765, 925
vendor supplied, compatibility of, 849
Verilog, 1026
lint-style checks, 915
List window

adding items to, 79
loading designs, interrupting, 1012
Local variables
print to Transcript, 126
log command, 421
log file
log command, 421
nolog command, 467
QuickSim II format, 1113
redirecting with -l, 1026
redirecting with -l, 1027
virtual log command, 880
virtual nolog command, 883
Log local variables, 126
lshift command, 425
lsublist command, 426
lteTime command, 626
ltTime command, 626

— M —

master slave library (SystemC), including, 577
mc_scan_plusargs, PLI routine, 1066
mem compare command, 427
mem display command, 428
mem list command, 431
mem load command, 432
mem save command, 436
mem search command, 439
memory window
add memory command, 84
adding items to, 84
memory, comparing contents, 427
memory, displaying contents, 428
memory, listing, 431
memory, loading contents, 432
memory, saving contents, 436
memory, searching for patterns, 439
merge
test-associated data, viewing, 245
merging coverage reports, 788
messages
echoing, 360
getting more information, 859
loading, disabling with -quiet, 924, 996
loading, enabling with -quiet, 765
-mfcu, 916

modelsim.ini

default VOPT behavior, 943

mulTime command, 626

multi-source interconnect de lays, 1031

— N —

name case sensitivity, VHDL vs. Verilog, 28

ncsim, one step simulation, 539

negative pulses

driving an error state, 1067

neqTime command, 626

nets

drivers of, displaying, 357

readers of, displaying, 554

stimulus, 395

values of

examining, 368

next command, 465

-no_risefall_delaynets, 1062

-nodebug argument (vcom), 760

-nodebug argument (vlog), 918

noforce command, 466

nolog command, 467

notepad command, 469

noview command, 470

nowhen command, 471

— O —

object_list_file, WLF files, 1117

onbreak command, 472

onElabError command, 474

onerror command, 475

onfinish command, 477

optimizations

disabling for Verilog designs, 922

optimizations

disabling for Verilog designs, 975

disabling for VHDL designs, 764

optimizing wlf files, 1119

order of events

changing in Verilog, 900, 903, 948

Order of precedence

compile, 572, 573, 903, 905, 906, 948, 949, 1018, 1058

— P —

pa autotestplan command, 478

parallel merge, vcover command, 798

parameters

using with DO files, 353

pathnames

in VSIM commands, 21

spaces in, 20

pause command, 486

PLI

loading shared objects with global symbol visibility, 1024

pop command, 487

Power Aware simulation, 1035

power report command, 496

power reset command, 499

Preoptimized Design Unit

report Verilog modules, 947

preference variables

WildcardFilter, 31

Preoptimized Design Unit

and SDF file, 1040

Preoptimized Design Unit, 995

main tain visibility into PDU, 995

printenv command, 501, 502

process report command, 503

profile clear command, 504

profile off command, 508

profile on command, 510

profile open command, 512

profile option command, 513

profile reload command, 515

profile save command, 521

profile summary command, 525

projects

override mapping for work directory with vcom, 579, 769

override mapping for work directory with vlog, 541, 933

propagatio n, preventing X propagation, 1033

property list command, 530

property wave command, 532

pulse error state, 1067

push command, 534

pwd command, 535

— Q —

questasim command, 536
QuickSim II logfile format, 1113
quietly command, 537
quit command, 538
qverilog command, 539

— R —

Radix
 color, 547
 example, 548
radix
 display values in debug windows, 543
 of signals being examined, 81, 97, 372
 user defined, 546
radix command, 543
Radix define command
 setting radix color, 547, 548
radix define command, 546
 fixed point radix, 546
 floating point radix, 546
radix list command, 551
radix name command, 552
range checking
 disabling, 762
 enabling, 765
readers command, 554
RealToTime command, 626
record field selection, syntax, 22
refresh, dependency check errors, 753, 908
refreshing library images, 765, 925
report command, 555
reporting
 processes in the Process Window, 503
 variable settings, 35
restart command, 558
restore command, 560
resume command, 561
right command, 562
run command, 565
runStatus command, 568

— S —

sc_stop()
 customizing simulator behavior, 1064
scaleTime command, 626

sccom command, 570
-scdpidebug command, 1075
scgenmod command, 581
-sclib com mand, 1075
-sclib command, 998
scope resolution operator, 24
scope, setting region environment, 366
SCV library, including, 577
SDF
 annotation verbose mode, 1042
 compiled SDF, 585
 controlling missing instance messages,
 998
 controlling missing instance messages,
 1041
 disabling individual checks, 616
 errors on loading, disabling, 1041
 warning messages, disabling, 1041
sdfcom command, 585
search command, 587
search libraries, 1026
searching
 binary signal values in the GUI, 52
 List window
 signal values, transitions, and names,
 42, 355, 680
 next and previous edge in Wave window,
 418, 562
 VHDL arrays, 40
 Wave window
 signal values, edges and names, 418,
 562
searchlog command, 590
see command, 593
seetime command, 594
setenv command, 595
shared objects
 loading with global symbol visibility, 961
shared objects
 loading with global symbol visibility, 1024
shift command, 596
shortcuts
 command history, 39
 command line caveat, 38
show command, 597

signals
 alternative names in the Wave window (-label), [95](#)
 attributes of, using in expressions, [43](#)
 breakpoints, [1104](#)
 combining into a user-defined bus, [98](#)
 drivers of, displaying, [357](#)
 environment of, displaying, [366](#)
 force time, specifying, [398](#)
 log file, creating, [421](#)
 pathnames in VSIM commands, [21](#)
 radix
 specifying for examine, [81](#), [97](#), [372](#)
 readers of, displaying, [554](#)
 stimulus, [395](#)
 values of
 examining, [368](#)

simulating
 design unit, specifying, [1012](#)
 simstats command, [598](#)
 simstatslist command, [601](#)
 simulating
 delays, specifying time units for, [35](#)
 design unit, specifying, [1012](#)
 ncsim style, [539](#)
 one step, [539](#)
 saving simulations, [421](#), [1050](#)
 stopping simulation in batch mode, [1111](#)

simulations
 saving results, [343](#), [344](#)

Simulator commands, [73](#)
simulator version, [1048](#), [1079](#)
simultaneous events in Verilog
 changing order, [900](#), [903](#), [948](#)

sml2ucdb command, [1142](#)
source annotation, [575](#)
spaces in pathnames, [20](#)
sparse memories
 listing with write report, [1132](#)

specify path delays, [1067](#)

stability checking
 disabling, [174](#)
 enabling, [175](#)

stack down command, [603](#)
stack frame command, [604](#)

stack level command, [605](#)
stack tb command, [606](#)
stack up command, [607](#)
startup
 alternate to startup.do (vsim -do), [1020](#)

status command, [608](#)

Std_logic
 mapping to binary radix, [52](#)

stop command, [611](#)

subTime command, [626](#)

suppress command, [612](#)

sv_reseed command, [614](#)

synthesis
 rule compliance checking, [749](#)

Syst emC
 specifying shared library path, command, [1075](#)

SystemC
 class and structure member naming syntax, [22](#)
 DPI, command for single-stepping across call boundaries, [1075](#)
 master slave library, including, [577](#)
 specifying shared library path, command, [998](#)
 verification library, including, [577](#)

SystemVerilog
 multiple files in a compilation unit, [916](#)

SystemVerilog
 enabling with -sv argument, [927](#)
 scope resolution, [24](#)

SystemVerilog assertions
 disabling, [1034](#)

— T —

tb command, [606](#), [615](#)
tcheck_set command, [616](#)
tcheck_status command, [621](#)

Tcl
 history shortcuts, [39](#)
 variable
 in when commands, [1108](#)

test management window
 adding UCDB files to, [90](#)

test-associated data
 viewing, [245](#)

testplan removal
 vcover remove command, 814

TFMPC
 disabling warning, 1063

time
 absolute, using @, 35
 simulation time units, 35

time col lapsing, 1050

Time commands, 626

time, time units, simulation time, 35

timescale directive warning
 disabling, 1063

timing
 disabling checks for entire design, 1034
 disabling checks, 920, 974
 disabling individual checks, 616
 status of individual checks, 621

title, Main window, changing, 1045

toggle
 reporting extended, 300, 819

toggle add command, 629

toggle coverage
 excluding signals, 633
 reporting, duplication of elements, 636
 reporting, ordering of nodes, 636

toggle disable command, 633

toggle enable command, 635

toggle report command, 636

toggle reset command, 638

toggle statistics
 enabling, 629
 reporting, 636
 resetting, 638

tr color command, 639

tr order command, 642

tr uid command, 644

transcribe command, 646

transcript
 clearing, 72
 redirecting with -l, 1026, 1027

transcript command, 647

transcript file command, 648

transcript path command, 650

transcript sizelimit command, 651

transitions, signal, finding, 418, 562

TreeUpdate command, 1129

TSCALE, disabling warning, 1063

TSSI, 1138

tssi2mti command, 675

typespec command, 676

— U —

-u, 932

UCDB coverage
 coverage attribute command, 255
 vcover attribute command, 772
 vcover dump command, 782
 vcover history command, 785

undeclared nets, reporting an error, 915

Unified Power Format, 993

unsetenv command, 679

up command, 680

UPF (Unified Power Format), 993

user-defined bus, 98

User-defined radix, 546

uvm call command, 683

uvm configtracing command, 686

uvm displayobjections command, 688

uvm findregisters command, 691

uvm findsequences command, 694

uvm handle command, 696, 698

uvm printconfig command, 700

uvm printfactory command, 703

uvm printstreams command, 706

uvm printtopology command, 708

uvm setverbosity command, 710

uvm sympath command, 712

uvm traceobjections command, 714

uvm uvmpath command, 716

— V —

-v, 932

v2k_int_delays, 1074

validTime command, 626

values
 describe HDL items, 348
 examine HDL item values, 368

variable settings report, 35

variables
 describing, 348
 referencing in commands, 35

value of
 changing from command line, 162
 examining, 368

vcd add command, 718

vcd checkpoint command, 721

vcd comment command, 722

vcd dumports command, 723

vcd dumportsall command, 726

vcd dumportsflush command, 727

vcd dumportslimit command, 728

vcd dumpportsoff command, 730

vcd dumpportson command, 731

vcd file command, 732

VCD files
 adding items to the file, 718
 capturing port driver data, 723
 converting to WLF files, 744
 creating, 718
 dumping variable values, 721
 flushing the buffer contents, 737
 generating from WLF files, 1116
 inserting comments, 722
 internal signals, adding, 719
 specifying maximum file size, 739
 specifying name of, 734
 specifying the file name, 732
 state mapping, 732, 734
 turn off VCD dumping, 741
 turn on VCD dumping, 742
 viewing files from another tool, 744

vcd files command, 734

vcd flush command, 737

vcd limit command, 739

vcd off command, 741

vcd on command, 742

vcd2wlf command, 744

vcom command, 747

vcover attribute command, 772

vcover dump command, 782

vcover history command, 785

vcover merge command, 788

vcover parallelmerge command, 798

vcover ranktest command, 803

vcover remove command, 814

vcover report command, 818

vcover testnames command, 843

vdel command, 847

vdir command, 849

Ve rilog
 \$finish behavior, customizing, 1064

vector elements, initializing, 162

vencrypt command, 852

vendor libraries, compatibility of, 849

Verilog
 \$finish behavior, customizing, 1064
 capturing port driver data with -dumports, 732

verror command, 859

version
 obtaining with vsim command, 1048
 obtaining with vsimcommands, 1079

vgencomp command, 861

VHDL
 arrays
 searching for, 40
 binding, ignore default, 756
 conditions and ex pressions, automatic conversion of H and L., 760
 field naming syntax, 22

VHDL identifiers, perserving case, 583

VHDL-2008
 package STANDARD
 REAL_VECTOR, 756

vhencrypt command, 863

Vi sualizer Debug Environment
 vopt, 954

view command, 866

viewing
 wav eforms, 1050

virtual count commands, 870

virtual define command, 871

virtual delete command, 872

virtual describe command, 873

virtual expand commands, 874

virtual function command, 875

virtual hide command, 879

virtual log command, 880

virtual nohide command, 882

virtual nolog command, 883

virtual region command, 885

virtual save command, 886
virtual show command, 887
virtual signal command, 888
vlib command, 894
vlog
 multiple file compilation, 916
vlog command, 898
vmake command, 936
vmap command, 939
vopt
 path separator, 944
vsim
 disabling internal setting of
 LD_LIBRARY_PATH, 1032
vsim comma nd, 1012

— W —

warning message
 6846, 788
WARNING[8], -lint argument to vlog, 916
warnings
 SDF, disabling, 1041
 suppressing VLOG warning messages, 974
 suppressing VCIM warning messages,
 763, 920, 974
 suppressing VLOG warning messages, 920
 suppressing VSIM warning messages,
 1063
watch window
 add watch command, 91
 adding items to, 91
watching signal values, 91
wave commands, 1082
wave create command, 1086
wave cursor commands, 1082
wave edit command, 1092
wave export command, 1095
wave import command, 1097
wave log format (WLF) file, 1050
 of binary signal values, 421
wave modify command, 1098
wave sort command, 1103
Wave window
 adding items to, 92
WaveActivateNextPane command, 1129
Waveform Comparison, 201

waveform editor
 creating waves, 1086
 editing commands, 1092
 importing vcd stimulus file, 1097
 modifying existing waves, 1098
 saving waves, 1095
waveform logfile
 log command, 421
waveforms
 optimizing viewing of, 1051
 saving and viewing, 421
WaveRestoreCursors command, 1129
WaveRestoreZoom command, 1129
when command, 1104
when statement
 time-based breakpoints, 1111
where command, 1112
wildcard characters
 for pattern matching in simulator
 commands, 30
WildcardFilter Preference Variable, 31
windows
 List window
 output file, 1129
 saving the format of, 1126
 Main window
 adding user-defined buttons, 76
 opening
 from command line, 866
 Wave window
 path elements, changing, 239
WLF files
 collapsing deltas, 1050
 collapsing time steps, 1050
 converting to VCD, 1116
 creating from VCD, 744
 indexing, 1119
 limiting size, 1051
 log command, 421
 merging, 1119
 optimizing, 1119
 optimizing waveform viewing, 1051
 repairing, 1124
 saving, 343, 344
 specifying name, 1050

wlfman command, [1117](#)
wlf2log command, [1113](#)
wlf2vcd command, [1116](#)
wlfman command, [1117](#)
wlfrecover command, [1124](#)
write cell_report command, [1125](#)
write list command, [1129](#)
write preferences command, [1130](#)
write report command, [1131](#)
write timing command, [1135](#)
write transcript command, [1137](#)
write tssi command, [1138](#)
write wave command, [1140](#)

— X —

X propagation
disabling X generation on specific instances, [616](#)
xprop assertlimit command, [1149](#)
xprop disable command, [1150](#)
xprop enable command, [1151](#)
Xpropagation
disabling for entire design, [1033](#)

— Y —

-y, [933](#)

— Z —

zoom
wave window
returning current range, [1084](#)

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, setup files 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. Except for Software that is embeddable ("Embedded Software"), which is licensed pursuant to separate embedded software terms or an embedded software supplement, 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 4.2); (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. BETA CODE.

- 3.1. 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.
- 3.2. 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.
- 3.3. 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 3.3 shall survive termination of this Agreement.

4. RESTRICTIONS ON USE.

- 4.1. 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. Except for Embedded Software that has been embedded in executable code form in Customer’s product(s), 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. Customer acknowledges that Software provided hereunder may contain source code which is proprietary and its confidentiality is of the highest importance and value to Mentor Graphics. Customer acknowledges that Mentor Graphics may be seriously harmed if such source code is disclosed in violation of this Agreement. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, disassemble, reverse-compile, or reverse-engineer any Product, or in any way derive any source code from Software that is not provided to Customer in source code form. 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.
 - 4.2. 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, or as permitted for Embedded Software under separate embedded software terms or an embedded software supplement. 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.
 - 4.3. Customer agrees that it will not subject any Product to any open source software (“OSS”) license that conflicts with this Agreement or that does not otherwise apply to such Product.
 - 4.4. 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.
 - 4.5. The provisions of this Section 4 shall survive the termination of this Agreement.
5. **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>.
 6. **OPEN SOURCE SOFTWARE.** Products may contain OSS or code distributed under a proprietary third party license agreement, to which additional rights or obligations (“Third Party Terms”) may apply. Please see the applicable Product documentation (including license files, header files, read-me files or source code) for details. In the event of conflict between the terms of this Agreement

(including any addenda) and the Third Party Terms, the Third Party Terms will control solely with respect to the OSS or third party code. The provisions of this Section 6 shall survive the termination of this Agreement.

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.** TO THE EXTENT PERMITTED 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. THIRD PARTY CLAIMS.

- 9.1. Customer acknowledges that Mentor Graphics has no control over the testing of Customer's products, or the specific applications and use of Products. Mentor Graphics and its licensors shall not be liable for any claim or demand made against Customer by any third party, except to the extent such claim is covered under Section 10.
- 9.2. In the event that a third party makes a claim against Mentor Graphics arising out of the use of Customer's products, Mentor Graphics will give Customer prompt notice of such claim. At Customer's option and expense, Customer may take sole control of the defense and any settlement of such claim. Customer WILL reimburse and hold harmless Mentor Graphics for any LIABILITY, damages, settlement amounts, costs and expenses, including reasonable attorney's fees, incurred by or awarded against Mentor Graphics or its licensors in connection with such claims.
- 9.3. The provisions of this Section 9 shall survive any expiration or termination of this Agreement.

10. INFRINGEMENT.

- 10.1. 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.
- 10.2. If a claim is made under Subsection 10.1 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.
- 10.3. 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; (h) OSS, except to the extent that the infringement is directly caused by Mentor Graphics' modifications to such OSS; or (i) infringement by Customer that is deemed willful. In the case of (i), Customer shall reimburse Mentor Graphics for its reasonable attorney fees and other costs related to the action.
- 10.4. THIS SECTION 10 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.

11. TERMINATION AND EFFECT OF TERMINATION.

- 11.1. 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.
- 11.2. Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination of this Agreement and/or any license granted under this Agreement, 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.
12. **EXPORT.** The Products provided hereunder are subject to regulation by local laws and European Union ("E.U.") 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, E.U. and U.S. government agencies. If Customer wishes to disclose any information to Mentor Graphics that is subject to any E.U., 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.
13. **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.
14. **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.
15. **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 15 shall survive the termination of this Agreement.
16. **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 or Japan, and the laws of Japan if Customer is located in Japan. 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, or the Tokyo District Court when the laws of Japan apply. Notwithstanding the foregoing, all disputes in Asia (excluding Japan) 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.
17. **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.
18. **MISCELLANEOUS.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements. Any translation of this Agreement is provided to comply with local legal requirements only. In the event of a dispute between the English and any non-English versions, the English version of this Agreement shall govern to the extent not prohibited by local law in the applicable jurisdiction. 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.