Mentor®
A Siemens Business

Questa® SIM Qrun User's Manual

Software Version 2020.4

Table of Contents

Chapter 1
The qrun Command Features and Operation 5

Major Features and Command Summary 7
Types of qrun Arguments 10
Option Groups 11
Re-Running Qrun. 12
Supported Languages and File Types 12
Modelsim.ini Option Sets Support for Qrun 13
Qrun Output Directory 14
Qrun Log Files 15

Chapter 2
Common qrun Command Arguments 17

Creating and Managing Libraries 17
Option Groups 20
Location of the Top Module 21
VHDL Compilation Order with the -autoorder Argument 21
Directing Tools to Run Specific Arguments 22
Debugging with the qrun -script and -env Arguments 23
Converting a Makefile or Transcript File to a qrun Command 24
Overriding Default Arguments 25
Qrun Command History. 25
Qrun Command Line Help 25

Chapter 3
Qrun Support for Simulation Methodology Flows 27

Pre-built Qrun Simulation Methodology Flows 27
Forcing a Verification Flow Step. 31
Incremental Compile 32
Qrun and Setting Up Regressions 32
Mixed Language Compilation and Simulation 33

Chapter 4
Qrun Command Reference 35

qrun 36

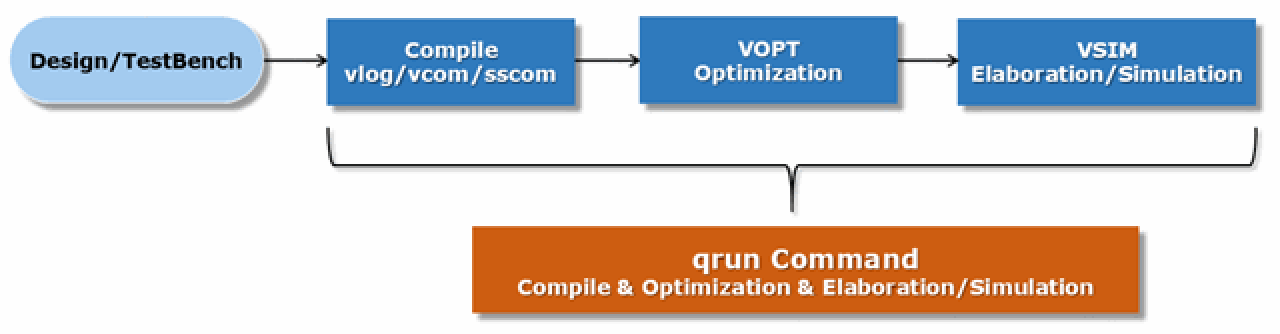
End-User License Agreement
with EDA Software Supplemental Terms

Chapter 1 The qrun Command Features and Operation

The qrun command simplifies the use of QuestaSIM by automatically combining the compile, optimize, and simulate functions into a single step.

Figure 1-1 shows the QuestaSIM verification flow and the commands defining each step, all being managed by the qrun command. With the qrun command, you can build a single step verification flow.You do not need to explicitly invoke the separate compile, optimize, and simulate commands (such as vlog, vopt, vcom, vsim) Knowledge of the two and three step flow is built into the command. The built in knowledge includes which underlying tool arguments to use.

Figure 1-1. QuestaSIM Verification Flow & qrun Command



In the discussion that follows, the term 'tool' is used to denote the compile, optimize, and simulate functions.

The basic syntax of the qrun command is:

qrun <myfiles> <optional arguments>

Note that arguments are optional. On invocation, qrun compiles, optimizes, and simulates "<myfiles>" without explicit direction (arguments) by you.

Consider the following Hello World SystemVerilog module. The file name is *hello.sv*

```
module IP;
initial begin
    $display("Hello world");
end
endmodule
```

Issue the following qrun command.

qrun hello.sv

Notice that there no arguments on the command line. You can see how qrun is managing the flow in the edited transcript below.

```
QuestaSim vlog Compiler 2019.04 Apr 11 2019
...
vlog hello.sv -work qrun.out/work -csession=incr -writesessionid
"+qrun.out/top_dus" -statslog qrun.out/stats_log
...
...
Errors: 0, Warnings: 0
QuestaSim vopt ...
vopt -findtoplevels qrun.out/top_dus -work qrun.out/work -o qrun_opt -
statslog qrun.out/stats_log
...
Analyzing design...
-- Loading module IP
Optimizing 1 design-unit (inlining 0/1 module instances):
-- Optimizing module IP(fast)
Optimized design name is qrun_opt
Errors: 0, Warnings: 0
Reading pref.tcl
# vsim -lib qrun.out/work -c -do "run -all; quit -f" qrun_opt -appendlog -
logfile qrun.log ...
...
# Loading sv_std.std
# Loading work.IP(fast)
# run -all
# Hello world
#  quit -f# *** Summary *********************************************
#     qrun: Errors:   0, Warnings:   0
#     vlog: Errors:   0, Warnings:   0
#     vopt: Errors:   0, Warnings:   0
#     vsim: Errors:   0, Warnings:   0
#   Totals: Errors:   0, Warnings:   0
```

If you use tool arguments on qrun command line, then the qrun command passes the arguments to the appropriate tool as in:

qrun hello.sv -o hello_opt

Qrun recognizes that the “-o hello_opt” argument is passed to the vopt command. The following transcript illustrates this.

```
vopt -o hello_opt -findtoplevels qrun.out/top_dus -work qrun.out/work -
statslog qrun.out/stats_log
```

You are *not* required to explicitly specify tool names when putting arguments on the qrun command line. The qrun command knows which arguments are associated with a tool without it being explicitly stated or written on the command line.

There are methods for you to explicitly call any tool arguments. These methods are documented in “Directing Tools to Run Specific Arguments” on page 22

Major Features and Command Summary **7**

Types of qrun Arguments 10
Option Groups 11
Re-Running Qrun 12
Supported Languages and File Types 12
Modelsim.ini Option Sets Support for Qrun. 13
Qrun Output Directory 14
Qrun Log Files 15

Major Features and Command Summary

The qrun command has these major features.

- All tool arguments and source files go on the same qrun command line.
  - All vlog, vopt, vsim arguments go on the qrun command line unmodified.
  - Supports two or three step flows.
- Simplifies the building of a mixed-language design by choosing the appropriate tool and arguments to compile for each file.
- Automatically detects file dependencies to compile files in the appropriate order.
- Maintains a history of previously executed commands, allowing you to browse command list. Next, you can re-run a line without having to specify all the previous arguments.
- Provides a way to output a shell script.
- Supports Pre-optimized Design Unit (PDU).
- Supports user over-rides of command defaults.
- Manages Libraries
  - No need to call vlib/vmap directly,
  - All libraries can be created inside or outside the standard qrun output directory.
- The qrun command can determine VHDL source file ordering.
- Provides pre-built flows for coverage, debug, UVM, Formal, and other flows. No need to construct arguments for multiple tools.

Table 1-1lists all the qrun arguments by functional category. Invoke qrun -help all on the command line to obtain the list.

Table 1-1. Qrun Argument Summary

qrun Arguments	**Usage Summary**
General Arguments	
-32, -64	Run in either 32-bit or 64-bit mode.
-verbose	Print additional output.
-version	Print the version of the qrun command.
Compiler/Optimization Controls	
-autoorder	An -autoorder compilation determines the proper order of VHDL design units, independent of the order that you listed the files on the command line.
-cdebug	Pass appropriate 'C' debugging flags to vlog and sccom.
-debug,<arg>,<arg>	Pass appropriate debugging flags to all tools
-defineall <macro[= value]>	Define <macro> for all compilers.
-uvm	Include the UVM library, optionally specified with uvmhome.
-uvmexthome <QUESTA_HOME/ questasim/verilog_src/ questa_uvm_pkg-1.2>	Compile the Questa UVM extensions into the work library.
-uvmhome <path-to-UVM-libraryUVM-version-string>	Specifies UVM library to use with -uvm.
Design units and libraries	
-clean	Remove 'qrun.out', or the directory specified with -outdir.
-cleanlib	Like -clean, but just remove the libraries.
-o <design_name>	Specify optimized output design name.
-outdir <filename>	Store libraries and data in 'filename' instead of the default, 'qrun.out'.
-top <design_unit>	Specify 'design_unit' as the top for simulation.
-work <library>	Compile into <library>.
Option and file specifications	
-defaultext=<tool>	Specify file extensions for default language
-<tool>.ext=[+][ext],[ext],[ext]...	Specify file extensions for the tool. A '+' adds to the current list.
-<tool>.options <option list for 'tool'> -end	Apply the included options to 'tool' (vlog, vopt, vsim, etc).
Flow Methodology Controls	

Table 1-1. Qrun Argument Summary (cont.)

qrun Arguments	Usage Summary
+UVM_TESTNAME=<testname>	Defines the +UVM_TESTNAME test name to be used for UVM designs.
-compile	Just compile the design. Do not optimize or simulate.
-coverage	Allows enabled coverage statistics to be kept.
-simulate	Force qrun to simulation without compilation and optimization.
-designfile <bin-filename>	Specifies a name for the Visualizer .bin file.
-gui[=interactive|postsim]	Specify mode in which to bring up the GUI. If 'postsim', qrun will not compile, optimize, or simulate.
-load_elab <filename>	Load simulation from previous elaboration
-noincr	Disable incremental compilation and optimization.
-optimize	Compile and optimize the design. Do not simulate.
-qwavedb=<options>...	Set options for Visualizer GUI.
-uvmcontrol=<arg>	Control specific UVM-aware debug options
visualizer[=desgin file name	Turn on Visualizer GUI. Default name for design file is design.bin.
Options to suppress or downgrade errors	
-error <msgNumber>[,<msgNumber>...]	Report the listed messages as errors.
-fatal <msgNumber>[,<msgNumber>...]	Report the listed messages as fatal.
-note <msgNumber>[,<msgNumber>...]	Report the listed messages as notes.
-permissive	Relax some language error checks to warnings.
-suppress <msgNumber/ msgGroup>[,<msgNumber/ msgGroup>...]	Suppress the listed messages using number or group.
-warning <msgNumber>[,<msgNumber>...]	Report the listed messages as warnings.
Scripts, environment, history, logging	
-env <filename>[(format)]	Store the commands and the current environment in 'filename', but do not execute the commands. The 'format' is in the style of printf with two '%s' specifiers, the first for the name and the second for the value.

Table 1-1. Qrun Argument Summary (cont.)

qrun Arguments	Usage Summary
-history=<arg>	Print previous commands. -history=N: Print command N. -history=N+redo: Rerun command N. -history=N+clear: Clear command N. -history=clear: Clear all commands. (N is a number, or a range of number. Ex:3-6)
-l <filename>	All output goes to 'filename'. -<tool>.log <filename>: Output for <tool> goes to 'filename'.
-log <filename>	All output goes to 'filename'. -<tool>.log <filename>: Output for <tool> goes to 'filename'.
-logfile <filename>	All output goes to 'filename'. -<tool>.log <filename>: Output for <tool> goes to 'filename'.
-script <filename>	Write the commands that would be executed by qrun into 'filename'. Qrun will exit without executing those commands.
-stats[=[+-]<args>]	Enables compiler statistics. <args> are all, none, time, cmd, msg, perf, verbose, list, kb.
Grouping files and options	
-filemap <vcom/vlog files, options> -end|-endfilemap	Compile the Verilog and VHDL files with the given options added.
-makelib <libpath[:logical]> <vcom/vlog files, options> -end|-endlib	Compile the Verilog and VHDL files into library 'libname'. The optional 'logical' name will be mapped to the path.
-precompile [libpath[:logical]]<vcom/vlog files, options> -end	Compile the Verilog and VHDL files with the given options before other files.
-reflib <library>	Adds a library to the search path. Similar to the vopt -L <library> switch.
SystemC Support	
-scgenmod <options> <design unit> <output file> -end	Create foreign module for SystemC.
-sclink <arguments> -end	Provide arguments for 'sccom -link'
-sysc	Compile C/C++ for SystemC

Types of qrun Arguments

There are two types of qrun arguments.

- Arguments specifically for qrun to execute such as -makelib and -script.

- Arguments which qrun passes to tools such as vlog or vopt. Qrun recognizes which tool to pass an argument to without you specifying the tool. For example the following command line contains the argument -nocvgzwopt.

  ```
  qrun ... -nocvgzwopt ...
  ```

  The qrun command passes the -nocvgzwopt argument to vsim without you explicitly specifying the vsim command as the command using the argument.

You can obtain command line help for any argument you specify without knowing which tool uses the argument. To use this feature append -help to the command line as in:

qrun -nocvgzwopt -help

The previous command returns the following:

```
vsim options:
------------------------------------------------------------------------
-nocvgzwopt     Enable sampling for zero weight covergroup
                items
```

The qrun -help command returns a summary of the -nocvgawopt argument and lists it as a vsim command option.

Option Groups

A qrun *option group*is a group of options applied to a given set of files and libraries.

The rules for constructing an option group are:

- Option groups begin with the operation you want to perform on a group of files or libraries.
- Next, a list of options.
- Finally an option set must be terminated by an -end statement.

Following are examples for the -makelib and -filemap option groups.

-makelib <library> <files> <options>   -end|-endlib
-filemap <options> <files>   -end|-endfile

Option sets allow you manage many arguments applied to a set of directories and files. The set of options you choose are only valid within the option group. You can have multiple option groups on the qrun command line

For example:

qrun -makelib mylib *.v -end

In the previous command, there are three sub-arguments to the -makelib option group. In turn, qrun invokes the following commands:

```
vlib -quiet qrun.out/mylib
vmap -c -quiet mylib qrun.out/mylib
```

The following example set shows how to use an option group to pass multiple arguments to the vlog command. Next, followed by the options for qrun to pass to the vopt command. The option gorup is terminated by -end.

- The option set begins with "-vlog.options" keywords directing qrun to pass the subsequent options to the vlog command.
- Next is the list of options: "-l vlog.log +define+A=345"
- Terminate the option set with the -end argument.

```
qrun test.sv -vlog.options -l vlog.log +define+A=345 -end
```

For additional information, consult the Managing Libraries"Creating and Managing Libraries" on page 17"Creating and Managing Libraries" on page 1.

Re-Running Qrun

If you re-invoke the same command with no source code or command line argument changes, then the qrun command skips compilation and optimization and invokes vsim. The resulting summary part of the transcript would be:

```
qrun hello.sv

...
# *** Summary *********************************************
#     qrun: Errors:   0, Warnings:   0
#     vsim: Errors:   0, Warnings:   0
#   Totals: Errors:   0, Warnings:   0
```

The qrun command puts all the results in a directory called *qrun.out*. Consult the subsection "Qrun Output Directory" on page 14 for details on the contents of the output directory.

If you want to rerun the entire compile, optimize, and simulate flow, use the -clean or -cleanlib arguments to delete and reset the work directories--and then re-invoke qrun hello.sv.

Supported Languages and File Types

The qrun command supports the following simulation languages:

- SystemVerilog/Verilog
- VHDL
- SystemC and C/C++ (DPI/PLI/FLI) source and compiled object files.

Table 1-2lists the files and tools recognized by qrun.

Table 1-2. Default File Extensions and Associated Types and Languages

Type	File Extension	Language
vlog	.v, .vp, .vs, .V, .VP, .VS	Verilog
vlog95	.v95, .V95, .v95p, .V95P	Verilog 95
svlog	.sv, .SV, .svp, .SVP, .svi, .svh, .vlib, .VLIB	SystemVerilog
vhdl	.vhd, .vhdp, .vhdl, .vhdlp, .VHD, .VHDP, .VHDL, .VHDLP	VHDL
vhdl	.vhcfg	VHDL Configuration
pslvlog	.pslvlog	PSL for Verilog
pslvhdl	.pslvhdl	PSL for VHDL
pslvlog	.pslvlog	PSL for Verilog
pslvhdl	.pslvhdl	PSL for VHDL
C	.c	C
cpp	.cpp, .cc, .cxx	C++
as	.s	Assembly
obj	.o	Object file
so	.so .sl	Shared object
spice	scs .sp	Spice
database	.bin .db	Database for Visualizer

Modelsim.ini Option Sets Support for Qrun

You can define option sets in the modelsim.ini file to create qrun shortcuts. Note, this feature is shared by QuestaSIM tools. It is not the same as qrun options sets which are type of command.

The first step is to edit the modelsim.ini file and define an option set. A modelsim.ini option set is defined as:

```
<option_name> = <options>
```

Once you define the option set in the modelsim.ini file, you reference it in qrun using the -optionset argument.The following example show how to define an option set called UVMDEBUG in the modelsim.ini file:

```
// In the modelsim.ini file.
UVMDEBUG = -uvmcontrol=all -msgmode both -displaymsgmode both -classdebug
-onfinish stop
VOPTDEBUG = +acc -debugdb
```

Note, once again, in the modelsim.ini file you are not required to explicitly state which tool will take the arguments: qrun does that for you. Pass the modelsim.ini option set definition to qrun using the -optionset argument:

qrun -optionset UVMDEBUG file1.sv file2.sv file3.vhd

Qrun Output Directory

By default, the qrun command dumps all output to the ./qrun.out directory. If the directory does not exist, qrun creates it.

You can change the location and name of the name of output directory as follows:

qrun -outdir <directory_name>

When you specify the -output argument, qrun creates a directory with the specified name, and uses that directory for all output.

The qrun.out directory contains these files and directories:

qrun test.sv

```
ls -al qrun.out
-rw-rw-r-- 1 mgc manuals  137 Mar 22 11:13 history
-rw-rw-r-- 1 mgc manuals    2 Mar 22 11:13 history.cnt
-rw-rw-r-- 1 mgc manuals  766 Mar 22 11:13 session
-rw-rw-r-- 1 mgc manuals  116 Mar 22 11:13 stats_log
-rw-rw-r-- 1 mgc manuals  100 Mar 22 11:13 top_dus
-rw-rw-r-- 1 mgc manuals    4 Mar 22 11:12 version
drwxrwxr-x 5 mgc manuals 4096 Mar 22 11:13 work/
```

The contents of the qrun.out directory are:

- history—The “history” file keeps track of the qrun commands you have executed. You can re-invoke previous commands with the argument history[=N[+redo]].
- history.cnt, stats_log, top_dus— General housekeeping files which you should not alter.
- session— The "session" file contains time stamp information to prevent unnecessary re-compilation and re-optimization.

- work— Directory contains all the libraries you specified with qrun. You can place the libraries anywhere on your network.

You can clear out the output files and directory using two arguments. They apply to the output files and directory of the current command line.

- -cleanlib [-outdir <output_directory>]. The -cleanlib option only cleans the specified library, and leaves behind the history and other files. If you do not specify an output directory name, then the qrun command deletes by default the qrun.out directory.
- -clean [-outdir <output_directory>]. Removes the entire qrun.out or the specified output directory. If you do not specify a output directory name, then the qrun command deletes by default the qrun.out directory.

You can first check if any of the reference or design libraries needs to be refreshed before executing any other commands.

```
qrun -refreshlibs -reflib lib1 -reflib lib2 -makelib lib3 file1.v file2.v -end top.vhd
```

Qrun Log Files

By default, qrun logs the transcript of any invocation in a file called qrun.log in the current working directory.

You can change the file name of the log file with the -log argument:

```
qrun -log <log_file_name>
```

None of the compile, optimize, simulate tools (vlog, vopt, vsim, etc.) produce log files by default. To create tool log files, use the argument <tool>.option, where <tool> is the name of the tool, and ".option" is a legal argument passed to the tool. For example:

```
qrun -vsim.log my_vsim_log
qrun -vlog.log my_vlog_log
```

In the this example, "log" is a legal argument to both the vsim and vlog commands.

You can put them all on the same qrun command line, as in:

```
qrun *.sv -vlog.log my_vlog_log -vopt.log my_vopt_log -vsim.log my_vsim_log
```

For additional information on using the <tool>.option argument, refer to "Directing Tools to Run Specific Arguments" on page 22.

Chapter 2
Common qrun Command Arguments

Creating and Managing Libraries 17

Option Groups 20

Location of the Top Module 21

VHDL Compilation Order with the -autoorder Argument 21

Directing Tools to Run Specific Arguments 22

Debugging with the qrun -script and -env Arguments. 23

Converting a Makefile or Transcript File to a qrun Command 24

Overriding Default Arguments. 25

Qrun Command History 25

Qrun Command Line Help 25

Creating and Managing Libraries

You can create and map work libraries using the -makelib argument. It is an option group argument. By default the work library is created in the ./qrun.out/work library.

The syntax for the creating a library is:

```
qrun -makelib <libpath[:logical]> <files>, [options] -end|-endlib
```

- The libpath argument is a Linux/Windows path to where qrun creates the library. The logical argument is the logical name of the library mapped to the physical library. If you do not specify a logical name, then qrun does not perform a logical mapping.
- The files argument specifies the source files compiled into the library.
- Optional arguments allow you to apply various tool arguments to the specified files.

Following is a simple example:

```
% qrun -makelib mylib *.v -end
```

The previous command invokes the vlib and vmap tools, as well as vlog, vopt, and vsim:

```
vlib -quiet qrun.out/mylib
vmap -c -quiet mylib qrun.out/mylib
```

Only the files and options within the -makelib -end command set are compiled by qrun into the library. You can terminated the -makelib option set with either the -end, -endlib text. If qrun encounters another -makelib argument, it terminates the previous one. You can have more than one -makelib -endlib command set on the qrun command line. You can use the -makelib in conjunction with other switches as in the following.

Using -makelib with Other Arguments

Using -makelib with other options gives you the flexibility to execute entire sets of source files. In the following example, compile file1.vhd and file2.vhd into library "mylib" with options -93. The -compile switch forces qrun to only perform the compile step and not optimization nor simulation.

```
qrun -compile -93 -makelib mylib file1.vhd file2.vhd -endlib ...
```

Compile files specified in mylib_files.txt into library "mylib" with options from -f file and -93.

```
qrun -compile -93 -makelib mylib -f mylib_files.txt -endlib ...
```

The following specifies a custom source file extension .myvhdl for the VHDL files specified in the -makelib collection libvhdl.

```
qrun -makelib libvhdl -vhdl_ext +.myvhdl bar.myvhdl -endlib tb_top.vhd
```

This example shows the specification of a physical library.

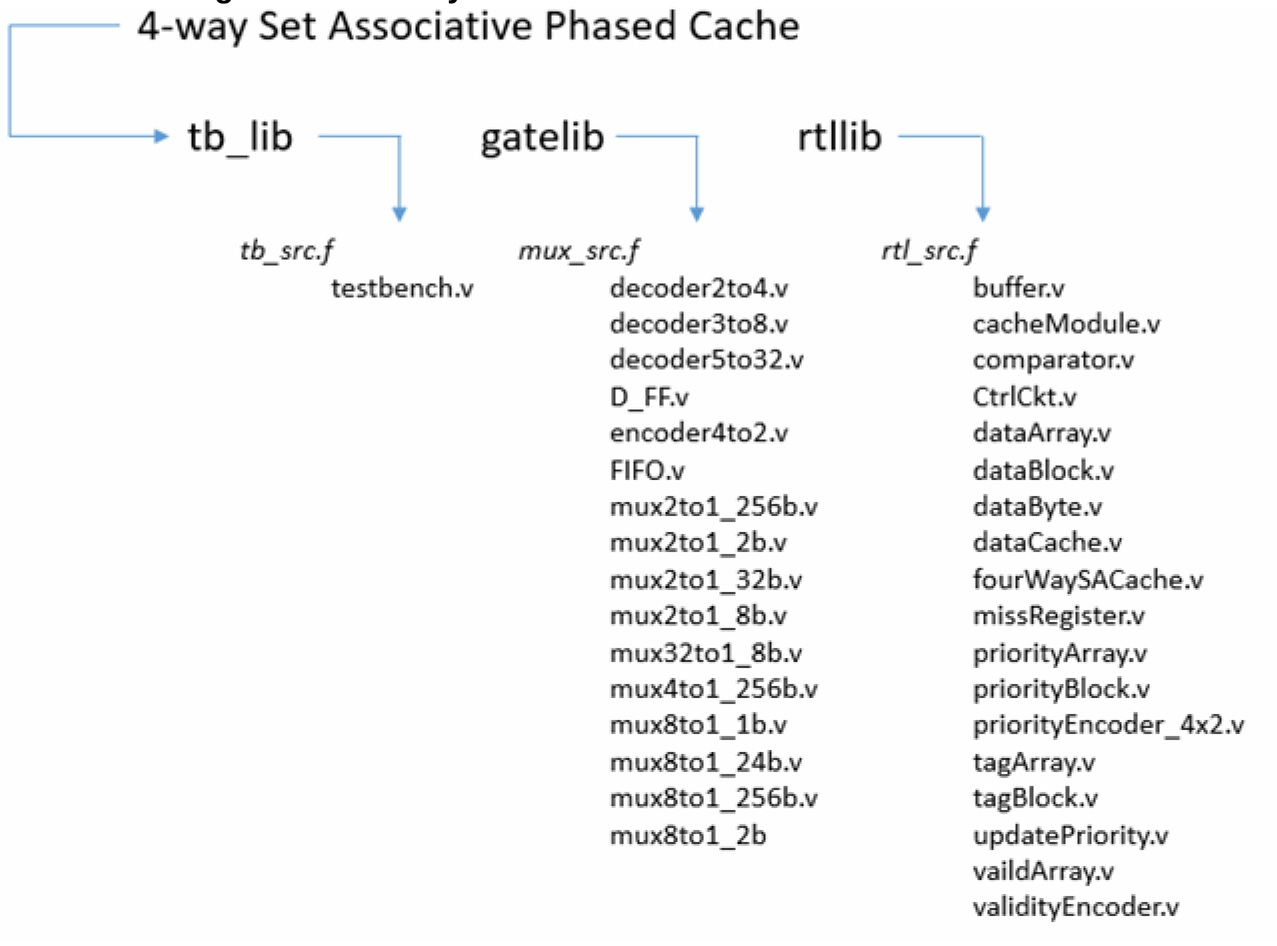
```
qrun -makelib /usr1/myarea/gates buf.v and2.v -endlib ...
```

When the qrun command line becomes more complicated, you can use the -f <filename> and -makelib arguments to ease complexity. The following example shows how to construct 4-way Set associative phased cache simulation.

Figure 2-1 shows the structure of the phased cache:

- The device has three libraries: tb_lib, gatelib and rtllib.
- All of the associated library source files have been put into their own separate *.f files: tb_src.f, mux_src.f, and rtl_scr.f.

Figure 2-1. Library Structure for a Associative Phase Cache



To explicitly build the libraries separately, use the qrun command as follows:

```
qrun -makelib rtllib    -f rtl.scr.f     -endlib
     -makelib gatelib  -f mux_scr.f  -endlib
     -makelib tb_lib    -f b_scr.f       -endlib
```

You put each one of the --makelib arguments into their own *.f file, which is named qrun.f shown as follows:

```
Contents of qrun.f
-makelib rtllib  -f rtl_src.f -endlib
-makelib gatelib -f mux_src.f -endlib
-makelib tb_lib  -f tb_src.f  -endlib
```

The final qrun command to build the libraries, optimize the files, and simulate is:

```
qrun -f qrun.f
```

The qrun command takes the device from compilation, optimization, to simulation without you specifying any tool argument:

```
# *** Summary *********************************************
#     qrun: Errors:   0, Warnings:   0
#     vlog: Errors:   0, Warnings:   0
#     vopt: Errors:   0, Warnings:   0
#     vsim: Errors:   0, Warnings:   0
#   Totals: Errors:   0, Warnings:   0
```

If you list the qrun.out directory, you can see that qrun has created directories for the three libraries.

```
.  ..  gatelib  history  history.cnt  rtllib  session  stats_log  tb_lib
top_dus  version  work
```

You can pre-compile files and reference libraries with -the reflib argument. Its syntax is:

qrun -reflib <library_name>

A related argument to -makelib is the -filemap argument. It allows you to specify how groups of files are to be compiled. Its syntax is:

qrun -filemap <options> <files> -endfile

Option Groups

A qrun *option group*is a group of options applied to a given set of files and libraries.

The rules for constructing an option group are:

- Option groups begin with the operation you want to perform on a group of files or libraries.
- Next, a list of options.
- Finally an option set must be terminated by an -end statement.

Following are examples for the -makelib and -filemap option groups.

-makelib <library> <files> <options>   -end|-endlib
-filemap <options> <files>   -end|-endfile

Option sets allow you manage many arguments applied to a set of directories and files. The set of options you choose are only valid within the option group. You can have multiple option groups on the qrun command line

For example:

qrun -makelib mylib *.v -end

In the previous command, there are three sub-arguments to the -makelib option group. In turn, qrun invokes the following commands:

```
vlib -quiet qrun.out/mylib
vmap -c -quiet mylib qrun.out/mylib
```

The following example set shows how to use an option group to pass multiple arguments to the vlog command. Next, followed by the options for qrun to pass to the vopt command. The option gorup is terminated by -end.

- The option set begins with "-vlog.options" keywords directing qrun to pass the subsequent options to the vlog command.
- Next is the list of options: "-l vlog.log +define+A=345"
- Terminate the option set with the -end argument.

```
qrun test.sv -vlog.options -l vlog.log +define+A=345 -end
```

For additional information, consult the Managing Libraries"Creating and Managing Libraries" on page 17"Creating and Managing Libraries" on page 1.

Location of the Top Module

The qrun command can usually locate the top level module for simulation even if you do not provide a name and location.

However, some test benches contain many top level modules or architectures, so it is good practice to manually specify a top module. To specify the top module, use the -top argument as in:

```
qrun -top top
```

Note

The qrun command can not determine the top level module when you use the -script argument. As a result. when you use the -script argument, explicitly specify the location of top.

VHDL Compilation Order with the -autoorder Argument

The qrun -autoorder argument determines the proper compilation order of VHDL design units, independent of the order that you listed the files on the command line. Without the argument, you must list VHDL files in their proper compilation order on the qrun command line.

There are two scopes which change the behavior of the -autoorder argument.

1. When you use the -autoorder argument within an option argument such as -makelib. In this case, the -autoorder argument only applies to any files listed in the options argument. In the following command example, qrun applies the -autoorder argument only to the files specified in the option group -makelib. For example:

   ```
   qrun -makelib ./mylib -autoorder  ./src/*.vhd  -endlib
   counter.vhd  test.vhd
   ```

   The source files counter.vhd and test.vhd must be in the proper VHDL compilation order. Within in the -makelib options argument, you can list VHDL files in any order and -autoorder properly orders their compilation.

2. When you use the -autoorder argument outside of any option argument on the main qrun command line, then the qrun command applies the -autoorder argument on all VHDL files regardless of where they appear on the command line. For example:

   ```
   qrun -autoorder -makelib ./mylib *.vhd -endlib
   flip.vhd counter.vhd
   ```

   In this case, all the files within the -makelib option and all files on the command line will have their compilation order determined by qrun.

Directing Tools to Run Specific Arguments

Qrun dispatches all arguments to the appropriate tools. There are circumstance, however, when you want to explicitly direct a particular argument to a specific tool. For example, vsim, vlog, and vopt do not create their own log files by default. You can use the tools to create their own specific log files.

The command to only generate a vsim log file is:

```
qrun -vsim.l my_sim_logfile ..."
```

In the previous command, qrun disptaches to vsim the argument “-l my_sim_logfile”.

The syntax to direct a singular argument to a specific tools is:

```
<-|+><tool>.<tool option>
```

where tool is vlog, vcom, vopt, vsim, or sccom.

If you only specify one tool, then the -end statement is not required. You can specify multiple argument associations, but the line must be terminated with an -end statement.

```
qrun -vlog.pedantic a.sv b.vhd
```

The syntax for applying multiple options for a tool is the following. It is also an option group.

```
-<tool>.options -opt1 -opt2 +op3 ... -end
```

This example shows how you can direct the vlog command to use any number of legal arguments:

qrun -vlog.options -pedantic - ... -end

Debugging with the qrun -script and -env Arguments

The qrun -script and -env allow you to capture all the commands invoked by qrun and all environment settings.

- -script <filename> — Produces a listing of every tool command and its arguments invoked by qrun for a particular command line invocation.
- -env <filename> — Does the same as the -script argument, but also lists all of the currently available window/terminal environment settings and variables.

Using the hello.sv example, you can get a listing of all the commands invoked by qrun:

qrun hello.sv -script csh.run -clean -top top

The qrun command puts the commands it invoked in csh.run. The -clean argument removes any libraries, so there will be no incremental compile. The -top argument locates the top module for qrun.

To send the contents of the csh.run file to the terminal, use the following command:

cat csh.run

```
vlib -quiet qrun.out/work
vlog hello.sv -work qrun.out/work
vopt top -work qrun.out/work -o qrun_opt
vsim -lib qrun.out/work -c -do 'run -all; quit -f' qrun_opt
```

The qrun command invoked vlib, vlog, vopt, and finally vsim.

The following qrun command implements a coverage flow for all Verilog files within a directory:

qrun *.v –coverage +cover=secb –script t.csh –top testbench

Append the -script <filename> argument to the command line to view what commands qrun invoked:

qrun *.v –coverage +cover=secb –top testbench -script myscript_output

The file myscript_output contains a listing of the commands invoked by qrun:

```
vlib -quiet qrun.out/work
vlog buffer.v cacheModule.v comparator.v CtrlCkt.v ...
vopt testbench -work qrun.out/work -o qrun_opt +cover=secb
vsim -lib qrun.out/work -c -coverage -do 'coverage save -onexit
cov_$Sv_Seed.ucdb'
-do 'run -all; quit -f' qrun_opt
```

Appending the -env myenv argument creates a myenv file which shows which tools were invoked by qrun, their arguments, plus any environment variables in the current shell:

```
REMOTEHOST=ORW-ESTVENS-920.wv.mentorg.comMANPATH=/usr/openwin/share/man:/
usr/man:/wv/dft/tools/share/man:/wv/dft/tools/ixl/man:/usr/wvcontrib/
man:/usr/wvcontrib/man/man1:/wv/let18/project/online/vendors/olias/olias/
man
VNCDESKTOP='orw-test-vm:50 '
SSH_AGENT_PID=2270
XDG_SESSION_ID=564
HOSTNAME=orw-test-vm
IMSETTINGS_INTEGRATE_DESKTOP=yes
SHELL=/bin/csh
SSH_CLIENT='147.34.71.109 52157 22'
....
....
vlib -quiet qrun.out/work
vlog buffer.v cacheModule.v comparator.v CtrlCkt.v ...
vopt testbench -work qrun.out/work -o qrun_opt +cover=secb
vsim -lib qrun.out/work -c -do 'coverage save -onexit cov_$Sv_Seed.ucdb'
-do 'run -all; quit -f' qrun_opt
```

Converting a Makefile or Transcript File to a qrun Command

You may want to convert existing makefiles to a qrun command for ease of use and future updates. You can also convert transcript files to a qrun command.

Prerequisites

- An existing makefile or QuestaSIM transcript file.

Procedure

1. Capture the output of “make –n <target>” to a qrun.f file.
2. Replace any instances of the vlib and vmap commands with –makelib … -lib arguments. Or remove them as qrun will create a work library.
3. Replace –l <logfile> with –tool.l <logfile> to generate tool specific log files.
4. Put “-top” before the top level DUT.

5. Remove the top level DUT from the vsim command line.
6. Remove –visualizer or –gui. Use instead the –visualizer and –gui=interactive arguments on the qrun command line.
7. Remove the tool names (vlog, vopt, etc) from the qrun command lines. Do not remove their arguments.

Overriding Default Arguments

You can override any default setting by explicitly specifying the argument on the qrun command line. You are not required to specify the tool. Qrun knows which tool to pass the new setting.

Qrun Command History

The qrun command keeps a history of all commands within a given qrun.out directory. In addition, you can use the -history argument to look up and replay commands.

The syntax of the -history swich is:

qrun -history<=Num>>+redo

The following shows a simple example of using the -history switch.

qrun -history

```
1 14:32 qrun a.vhd
2 14:50 qrun b.vhd c.sv
3 16:20 qrun -mfcu -93 +acc -o opt -c a.sv b.sv c.vhd
```

Run qrun -history to show the second executed command.

qrun -history=2

```
2 14:50 qrun b.vhd c.sv
```

Re-run the second command previously executed. Using the previous example, qrun executes “qrun b.vhd c.sv”.

qrun -history=2+redo

Qrun Command Line Help

Qrun contains an extensive command line help feature.

The basic syntax for the help feature is:

qrun -help [option]

The qrun -help argument is organized to provide help on various categories, individual arguments, and the entire set of arguments for qrun.

To get a summary of all the qrun commands along with their categories, issue the following qrun command.

qrun -help all

The following command shows how to list all the categories of command types:

qrun -help

Produces:

```
General          List most common general options
Compiler         List options for compiler, and optimizer controls
Design           List options for design units, and libraries
File             List options for file and option specifications
Flow             List options for Flow (Coverage, PA, Debug, ...)
Messages         List all warn, error, note, fatal messages options
Script           List options for scripts, environment, history, and log
Subgroup         List options for handle grouping files and options
SystemC          List options supporting for SystemC
```

Get help listing of all commands within a Flow category:

qrun -help Flow

One feature of -help is that you can get help on any argument on the qrun command line. You may omit source files or libraries. The following shows how to get help on two specific arguments which are -93 and -coverage:

qrun -93 -coverage -help

The previous command issues the following. Note that the qrun help identifies every tool associated the switch.

```
vcom options:
-------------------------------------------------------------------------
-93                 Enable support for VHDL 1076-1993 in VHDL-style.
                   PSL files specified with -pslfile_vh <file>vsim options
-------------------------------------------------------------------------
-coverage          Allows enabled coverage statistics to be kept
```

Chapter 3
Qrun Support for Simulation Methodology Flows

Pre-built Qrun Simulation Methodology Flows 27
Forcing a Verification Flow Step 31
Incremental Compile 32
Qrun and Setting Up Regressions 32
Mixed Language Compilation and Simulation 33

Pre-built Qrun Simulation Methodology Flows

The qrun command contains a number of pre-defined flows that allow you to quickly implement various simulation methodologies. With these predefined flows, you are not required to put together switches for tools such as vlog or vsim. The qrun command applies the arguments to the underlying tools to create the methodology flow.

Figure 3-1 shows different sorts of verification methodologies that teams typically implement. The qrun flows allow an easy way to immediately use the underlying tools without complex setup.

Figure 3-1. Types of Verification Methodologies

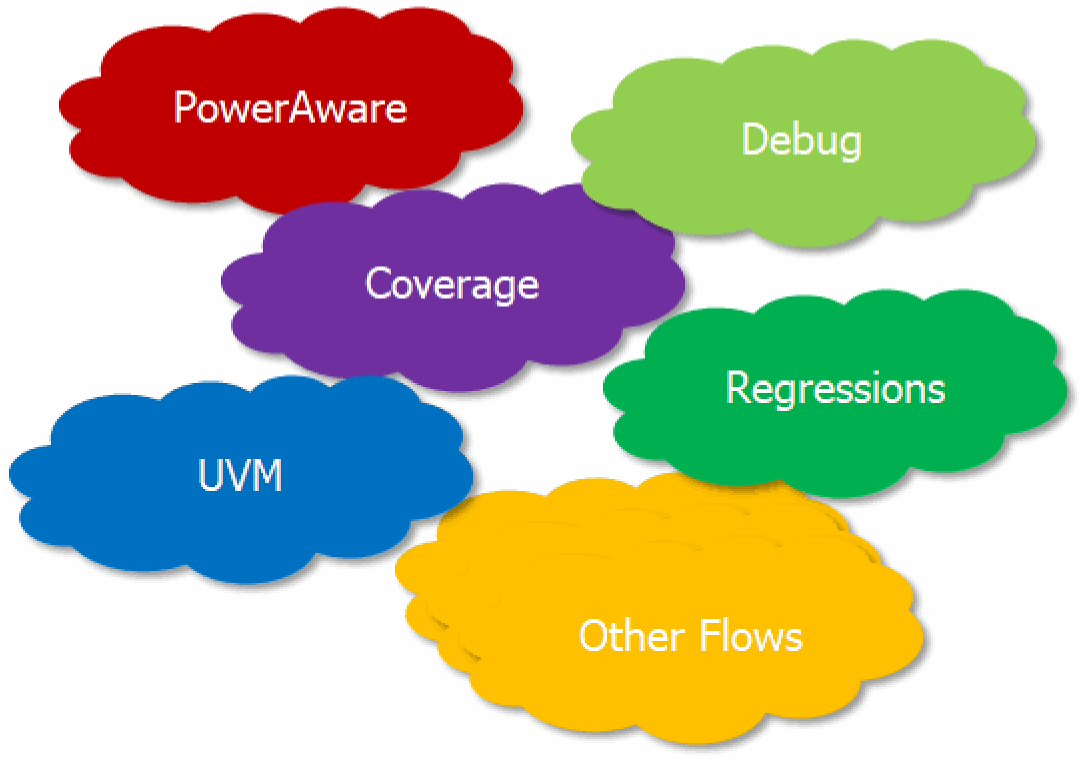


Table 3-1 lists the available pre-defined flows.

Table 3-1. Qrun Supported Flows

Flow Information	Flow switch and options
GUI flow. • The –gui switch tells qrun to bring up the GUI in interactive or postsim mode. • –gui=postsim only brings up the GUI in postsim mode. Qrun will not compile, optimize, or simulate the design. • The default is to bring up the Questa classic gui in interactive mode. • If -visualizer is on the command line, qrun opens the Visualizer GUI with the appropriate database files.	-gui[=int[eractive]|postsim]
Coverage Flow. • Collects coverage and creates a UCDB file. • Allows enabled coverage statistics to be collected.	-coverage

Table 3-1. Qrun Supported Flows (cont.)

Flow Information	Flow switch and options
Debug Flow. • Launches QuestaSIM GUI or Visualizer. Passes appropriate debugging flags to all tools • The –debug switch adds different options to the tool command lines based on which GUI is selected.	-debug[,<livesim|cell|etc>+]
C (language) Debug Flow. • Passes appropriate 'C' debugging flags to vlog. • Add –g to the C/C++ compilers and –ccflags –g to all the vlog command lines, and –g sccom command lines.	-cdebug
UVM Flow. • Allows you to select which UVM version and/or source code to use. • Automatically includes required +incdir switches. Compiles uvm_dpi.cc.	UVM
Visualizer Flow. • The default GUI is the Questa Classic GUI. If the –visualizer switch is used on the qrun command line, qrun uses the options associated with Visualizer in both postsim and interactive modes.	-visualizer[=designfile_name]

You can invoke either the QuestaSIM Classic GUI or Visualizer in interactive or postsim mode. Qrun automatically sets all switches that are necessary to capture the relevant debug information. In interactive mode, the GUI will come up at the command prompt so you can log any necessary signals and set breakpoints.

In the following command, the qrun command invokes the Visualizer tool in post-simulation mode with all the generated Visualizer debug databases.

```
qrun test.sv gui= postsim –visualizer
```

The result of the previous command is to invoke Visualizer on the database (.bin) file produced by vsim.

You can override the default arguments qrun passes to the commands by invoking explicit arguments. Note in the following override example you are not required to specify the tool.

```
qrun –designfile mydesign.bin -qwavedb=+memory –visualizer –gui=interactive
```

To enable the coverage flow with qrun, use the following command:

```
qrun test.sv -coverage
```

Note from the transcript below the tool directives which implement coverage. You are not required to supply the arguments on the command line.

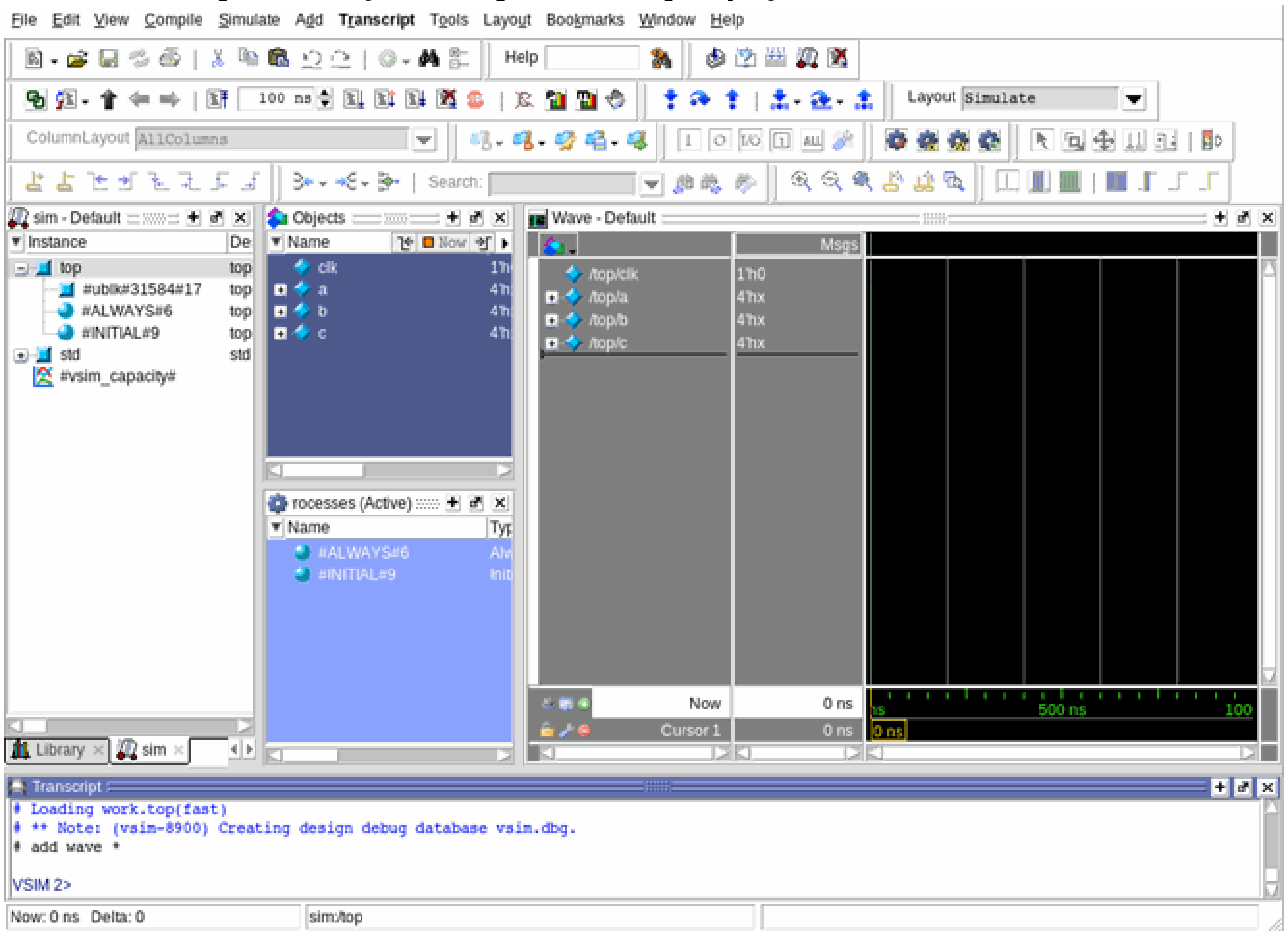
```
...
vsim -lib qrun.out/work -c -do "coverage save -onexit cov_$Sv_Seed.ucdb" -
do "run -all; quit -f" qrun_opt -appendlog -logfile qrun.log
....
# Loading sv_std.std
# Loading work.top(fast)
# coverage save -onexit cov_$Sv_Seed.ucdb
# run -all
# a: 1, b:15, c: 5
# a: 5, b: 8, c: 5
...
# ** Note: $finish    : test.sv(26)
#    Time: 190 ns  Iteration: 1  Instance: /top
# Saving coverage database on exit...
...
```

Invoke the QuestSIM Classic GU

qrun test.sv -gui

The previous command produces the following:

Figure 3-2. Qru Debug Flow Brings Up QuestaSIM Clasical GUI



```
# Loading work.top(fast)
# ** Note: (vsim-8900) Creating design debug database vsim.dbg.
# add wave *

VSIM 2>
```

Now: 0 ns Delta: 0   sim:/top

Forcing a Verification Flow Step

Qrun gives you the option of forcing three steps of the verification flow: compile, optimization, and simulation.

Prerequisites

- A legal SystemVerilog file. Name it test.sv.

Procedure

1. Issue the following command to force qrun to stop after the compile step.

   qrun test.sv -compile

2. Issue the following command to force qrun to stop after vopt executes.

   qrun test.sv -optimize

3. Issue the following command to force qrun to only run a simulation. Qrun does not check to determine if another recompile or re-optimize is required.This also forces qrun to simulate even if switches or source files have changed.

   qrun test.sv -simulate

Incremental Compile

When you re-invoke the qrun command, it checks to see if it needs to re-compile or re-optimize the design.

The qrun command will go directly to simulation if:

- The content of the source files have not changed since the last run. This includes source files supplied with -y and -v arguments.
- The order of the source files specified on the command line, including the order of -v and -y arguments, has not changed
- Any argument that impacts compilation or optimization has not changed. For example, adding -93 causes a recompile. However, adding or modifying -do or -sv_seed, does not cause a recompile.

Qrun and Setting Up Regressions

Using arguments for controlling qrun's invocation of the compile, optimize, and simulate steps, you can use qrun to run regression tests.

In the following examples, all source files and compile, optimization, and simulations switches are contained in a *qrun.f* file. Use the –snapshot argument to name the compile/optimize output. It is also used to indicate which snapshot to simulate.

One method to using qrun to control regression tests is as follows:

1. Compile and optimize only; and, explicitly name the output directory. Use the -optimize argument to force qrun to only compile and optimize.

```
# compile and optimize
qrun -f qrun.f -optimize -snapshot <snapshot name>
# this compiles and optimizes the design and testbench
```

2. Simulate with qrun using individualized test arguments. Use the -simulate argument to direct qrun to only simulate your individual tests.

```
# run simulations
qrun -simulate -snapshot <snapshot name> [-sv_seed <seed value>]
[+UVM_TESTNAME=<testname>]
```

An extended example follows:

```
# Clean up qrun output files
qrun -clean

# Compile and optimize the design and testbench
qrun -f qrun.f -optimize -snapshot rtl_sims -nosva +notimingchecks
+nospecify

# Run the simulations
qrun -simulate -snapshot rtl_sims +UVM_TESTNAME=test1
qrun -simulate -snapshot rtl_sims +UVM_TESTNAME=test2 -sv_seed  500
qrun -simulate -snapshot rtl_sims +UVM_TESTNAME=test2 -sv_seed   25
qrun -simulate -snapshot rtl_sims +UVM_TESTNAME=test2 -sv_seed 1255
```

It is not necessary for you to break the compile/optimize and simulate flow into two separate steps. The previous flow uses the –simulate argument to skip the check to determine if the source code should be compiled and optimized. This ensures that no compilation occurs if a source files is accidentally modified while the regressions are running.

Because qrun does incremental compilation, the following flow example will also compile and optimize once, then simulate without recompiling. In the following example, qrun –clean will remove all qrun generated files. There will be some small overhead for the incremental compilation check. Note the following:

- The first “qrun –f qrun.f …” command compiles, optimizes, and simulates test1.
- The following “qrun –f qrun.f …” commands only run the simulation with the specified UMV testname and SV seed value.

```
# Clean up qrun output files
qrun -clean# Run the simulations
qrun -f qrun.f +UVM_TESTNAME=test1
qrun -f qrun.f +UVM_TESTNAME=test2 -sv_seed  500
qrun -f qrun.f +UVM_TESTNAME=test2 -sv_seed   25
qrun -f qrun.f +UVM_TESTNAME=test2 -sv_seed 1255
```

Mixed Language Compilation and Simulation

The qrun command compiles VHDL-Verilog/SystemVerilog and VHDL-SystemC designs with no package sharing in the following order:

1. All SystemC files.
2. All Verilog and SystemVerilog files.
3. All VHDL files.

This ensures a VHDL design can support entity instantiation of design units of the other language.

If you need Verilog or VHDL files compiled at an earlier point, you can use the -precompile argument whose syntax is:

```
-precompile [<library>] <files and options> [-end]
```

The specified files will be compiled before the normal order of compilation.

In this example, qrun first compiles the VHDL file test1.vhd with the argument -mixedsvvh. Note that you can also specify entire libraries for pre-compilation.

```
qrun -precompile test1.vhd -mixedsvvh  -end test1.sv
```

You can define macros available for use by all C, C++, SystemC, Verilog and SystemVerilog programs using the -defineall argument, whose syntax is:

```
qrun -defineall <macro_name>[=<macro_value>]
```

For example:

```
qrun -defineall foo
qrun -defineall MAXWIDTH=32
```

Chapter 4
Qrun Command Reference

qrun **36**

qrun

Compiles, optimizes and simulates a design by taking all input files and arguments together and automatically executing the correct tool and command-line arguments for each file.

Syntax

This subsection lists qrun command arguments by functional category.The Arguments section groups the argument descriptions in alphabetical order.

qrun [files] [options]

General: most common options

[-32 | -64] [-f | F <optionfile>] [-help [options|topic]] [-verbose] [-version]

Compiler Control: options for compiler, and optimizer controls

[-autoorder] [-cdebug] [-debug, <arg>, <arg>, ...] [-defineall <macro[= value]>] [-uvm] [-uvmexthome <QUESTA_HOME/questasim/verilog_src/questa_uvm_pkg-1.2>]

Design units and libraries

[-clean] [-cleanlib] [-o <design_unit>] [-makepdu <pdu_name> [/path/to/lib.]<du_name> <options> -end]-o <design_unit> [-outdir <directory>] -snapshot <snapshot_name> [-top <design_unit>]

File and Tool Association

[-defaultext=<tool>] [-<tool>.ext=[+][ext],[ext],[ext]...] [<-|+><tool>.<tool option>] [-<tool>.options -opt1 -opt2 +op3 ... -end]

Flow Controls

[+UVM_TESTNAME=<testname>] [-compile][-coverage] [-debug, <arg>, <arg>, ...] [-designfile <bin-filename>] [-gui[=interactive | postsim]] [-load_elab <filename>] [-noincr] [-optimize] [-qwavedb=<options>] [-uvmcontrol]-visualize

Formal/CDC

[-autocheck] [-cdc][-covercheck] [-cuname <file>] [-fx] [-genucdb [filename]] [-goal] [-hcdc] [-hrdc] [-lint] [--methodology] -od] [-propcheck] [-protocol] [-rdc [-rdc] [-report_clock] [-reset] [-reset] [-sdc [-simulate] [-timeout] [-xcheck]

Messages

[-error <msgNumber>[,<msgNumber> ...]] [-fatal <msgNumber>[,<msgNumber> ...]] [-note <msgNumber>[,<msgNumber> ...]] [-permissive] [-suppress <msgNumber/msgGroup>[,<msgNumber/msgGroup>...]] [-warning <msgNumber>[,<msgNumber> ...]]

Script, Environment, History, and Logging

[-env <filename> [<format>]] [-history=<arg>] [[-log | -l | -logfile] [<filename>]] [-script <filename>] [-stats[=[+-]<args>]]

Subgroup options for handle grouping files and options

[-filemap <vcom/vlog files, arguments> -endfilemap] [-makelib <libpath[:logical]> <vcom/vlog files, options> -end|-endlib] [-precompile [libpath[:logical]] <vcom/vlog files, options> -end] [-reflib <library>] [-refreshlibs]

SystemC

[-scgenmod <options> <design unit> <output file> -end] [-sclink <arguments> -end] [-sysc <sccom source files> <sscom options> -end]

Arguments

- -32 | -64

  Run in 32 or 64 bit mode.

- -autocheck

  Enable autocheck for Formal flow.

- -autoorder

  Perform an -autoorder compilation on all -makelib subgroups.The argument compiles VHDL files in the proper order regardless of the order specified on the command line. Source files outside -makelib subgroups can be specified in any order.

- -cdc

  Enable cdc flow.

- -cdebug

  Pass appropriate “c” debugging flags to vlog and sccom.

- -clean

  Removes the directory *qrun.out* or the directory specified with the -outdir argument. The -clean argument cleans the content of the output sub-directory related to this run of qrun. The sub-directory will be <output_dir>/<platform>_<version>.

- -cleanlib

  Removes libraries. The -cleanlib argument only cleans the library and leaves behind the history and other files.

- -compile

  Compile the design without optimization or running a simulation.

- -covercheck

  Enable covercheck flow for Formal flow.

- -coverage

  Enables the coverage flow.

- -cuname <file>

  Specify CU name

- -debug, <arg>, <arg>, ...

  Pass appropriate debugging flags to all tools.

- -defaulttext=<tool>

  Specify file extensions for default language.
- -defineall <macro[= value]>

  Define <macro> for all compilers. The -defineall argument provides a way to provide a single definition of a macro to be available in all of these languages: - C, C++, SystemC, Verilog and SystemVerilog.
- -designfile <bin-filename>

  Specifies the name of the Visualizer design.bin file.
- -env <filename> [<format>]

  Store the commands and the current environment in *filename*. The format argument is in the style of printf with two '%s' specifiers. The first '%s specifier is for the name, and the second for the value. You must put quotes around filename and format. For example:

  ```
  qrun test.sv -env "myenv(setenv %s %s )"
  ```
- -error <msgNumber>[,<msgNumber> ...]

  Reported the listed messages as errors.
- -f | F <optionfile>

  This '-f file' argument causes qrun to read arguments from the specified file. The syntax of the file is same as described in the QuestaSIM Reference Manual.
- -fatal <msgNumber>[,<msgNumber> ...]

  Report the listed messages as fatal.
- -filemap <vcom/vlog files, arguments> -endfilemap

  Compile the Verilog and VHDL files with the given arguments. You can terminate the argument list using -end also. You cannot use +define statements using -filemap.
- -fx

  Specify FX generation (cdc generate fx).
- -genucdb [filename]

  Specify ucdb file name
- -goal

  Specify goal (ignore goal if methodology not specified)
- -gui[=interactive | postsim]

  Specific mode in which to bring up the GUI. If postsim, then qrun will not compoile, optimize, or simulate.
- -hcdc

  Specify abstract model generation.

- -hrdc

  Specify abstract model generation (-hrdc).

- -history=<arg>

  Prints out or re-executes a previously used command. Without the use of arguments, the qrun command returns all previous commands. Qrun will store a 'history' file in its output directory. This file will be maintained containing the list of all 'qrun' commands that have been executed. A 'logs' directory will contain an output file from each run, named to correspond to a history item. Invoking '-history' with a command-line argument will list the history providing an argument to redo the item.

  In the following argument list, N represents a number or a range of numbers.

  -history=<N> — Prints command N.

  =<N>+redo — Reruns the command N.

  -history=N+clear — Clear command N

  -history=clear — Clear all commands.

- -help [options|topic]

  Lists all qrun command arguments if you do not specify any arguments.

  You can specify that the qrun command give you available information on either topics or a tool.

- -lint

  Enable Lint flow.

- -load_elab <filename>

  Load simulation from previous elaboration.

- [-log | -l | -logfile] [<filename>]

  Prints a log to either the terminal or filename. Both “l” and “log” aliases for logfile.

- -noincr

  Disable incremental compilation and optimization.

- -makelib <libpath[:logical]> <vcom/vlog files, options> -end|-endlib

  Compile Verilog and VHDL files into library libpath. The optional logical name is mapped to the path. The library name can be the physical or the logical library name. A makelib argument is terminated with another sub-list argument like -makelib or -filemap or by using the -end argument. The command creates the library if it does not exit.

  The -stat arguments is not allowed within -makelib

- -makepdu <pdu_name> [/path/to/lib.]<du_name> <options> -end

  Optimize the desired design units into a PDU called 'output'.

  - The “lib.” prefix to <du_name> is the library to read the DU from.

        - lib can be a logical name or a physical path to the library
    - Vopt requires that the PDU be written into the same library that the DU is read from
        - The PDU will use the value of "lib." to determine the library that is is written to.
    - makepdu/end commands can be mixed with other non-pdu compile, optimization, and simulation commands.
        - qrun will create the PDU's in separate vopt invocations .
        - A toplevel vopt will be run to bring all the PDU's and non-pdu DU's together.
        - Subsequent qrun invocations will skip the PDU generation steps if there are no source/switch changes.

- --methodology

  Specify methology and goal.
- -note <msgNumber>[,<msgNumber> ...]

  Report the listed messages as notes.
- -o <design_unit>

  Specify optimized output design name.
- -od

  Specify output directory for qverify.
- -optimize

  Compile and optimize the design. Does not simulate the design. When you specify this argument, qrun compiles the source files provided on the command line, and then the design by calling vopt. If the design has been optimized previously, then qrun will call vopt only if the design or the arguments have not changed since the last optimize step.
- -outdir <directory>

  Store libraries and data in the specified directory instead of the default directory *qrun.out*. By default qrun creates an output directory *qrun.out* if one does not exist in the current directory. When you use the -outdir argument, qrun creates and uses the output directory specified by the -outdir argument.
- -permissive

  Relaxes some language error checks to warnings.
- -precompile [libpath[:logical]] <vcom/vlog files, options> -end

  Compile the Verilog and VHDL files with the given arguments before other files.
- -propcheck

  Specify protocol generation (cdc generate protocol).

- -protocol

  Enable propcheck flow for Formal flow.

- -qwavedb=<options>

  Set options for Visualizer GUI.

- -snapshot <snapshot_name>

  Specify snapshot (optimized design) name. Does the same as the -o switch but -snapshot is more descriptive.

- -rdc

  Enable RDC flow.

- -refreshlibs

  First checks if any of the reference or design libraries needs to be refreshed before executing any other commands.

- -reflib <library>

  Add the specified library to all search paths. This argument would be required if qrun creates and manages all the libraries in the design. Files can be precompiled into a library with -compile and -makelib argument and then referred to in a subsequent qrun command with the -reflib argument. The reflib argument adds the library to the list of libraries to search for a given design unit.

- -report_clock

  Specify clock reporting only.

- -reset

  Specify clock reporting only (-report_reset).

- -scgenmod <options> <design unit> <output file> -end

  Create a foreign module for SystemC.

- -script <filename>

  Write the commands that would be executed by qrun into 'filename'. Qrun will exit without executing those commands.

- -sclink <arguments> -end

  Provide arguments for 'sccom -link' command.

- -sdc

  Specify SDC file (sdc load).

- -simulate

  Only simulate the design, without compiling or optimizing.

- -stats[=[+-]<args>]

  Enables or disables compile stats. Args can have the following values.

=[+-] — Controls activation of the feature or mode. The plus character (+) enables the stat, and the minus character (-) disables the state.

all — Display all statistics features (cmd, msg, perf, time). Mutually exclusive with none argument. When specified in a string with other arguments, 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 argument. When specified in a string with other arguments, none is applied first.

perf — Display time and memory performance statistics.

time — (default) Display Start, End, and Elapsed times.

- -suppress <msgNumber/msgGroup>[,<msgNumber/msgGroup>...]

  Suppress the listed messages with either message number or group.

- -sysc <sccom source files> <sscom options> -end

  Compile SystemC source files with the specified options. The -sysc/-end switches are required for systemC source files and options. This is how qrun distinguishes SystemC C/C++ files from DPI or VPI source files. The -end is optional. The command set can also be terminated with -makelib or -filemap.

- -timeout

  Specify time out.

- -<tool>.log <filename>

  Print the log for a particular tool to specified filename.

- <-|+><tool>.<tool option>

  Allows you to associate a single argument to a specific tool.

  -|+ — If the tool option is a plusargs (starts with a +), the invocation is +<tool>.<option>. For example: +vlog.acc.

  If it starts with a '-', then the invocation is -<tool>.<option>. For example: -vlog.fsmdebug.

- -<tool>.options -opt1 -opt2 +op3 ... -end

  Associates a list of options with a tool. You can specify multiple arguments with a tool, but you must terminate the command line with a “-end” statement. The tool argument maybe vlog, vopt, vsim or any other supported tool. There is no plusargs associated with <tool> with the list of options

- -top <design_unit>

  Specifies design_unit as the top for simulation. The top level design units of for VHDL and SystemC are not automatically detected. As a result, VHDL and SystemC top must be specified by the -top argument.

All compiled but unreferenced Verilog and SystemVerilog modules will be treated as top level instances for optimization and simulation, unless you use the -top argument. They also become top level modules unless you specify -svext=uslt with the qrun command.

- -<tool>.ext=[+][ext],[ext],[ext]...

  Associates a file type with a tool or language. The qrun command supports several default suffixes listed in the following table.

  You can override these using the -<tool>.ext argument, along with an '=' to replace the list, or a '+' to add to the list. For example:

```
qrun -vlog.ext=.v,.vv,.vh :Replace Verilog suffixes list
qrun -vhdl.ext=+.vhdh,.xyz : Add '.vhdh'& .xyz to VHDL list
```

Table 4-1. Default File Extensions and Associated Types and Languages

Type	File Extension	Language
vlog	.v, .vp, .vs, .V, .VP, .VS	Verilog
vlog95	.v95, .V95, .v95p, .V95P	Verilog 95
svlog	.sv, .SV, .svp, .SVP, .svi, .svh, .vlib, .VLIB	SystemVerilog
vhdl	.vhd, .vhdp, .vhdl, .vhdlp, .VHD, .VHDP, .VHDL, .VHDLP	VHDL
vhdl	.vhcfg	VHDL Configuration
pslvlog	.pslvlog	PSL for Verilog
pslvhdl	.pslvhdl	PSL for VHDL
pslvlog	.pslvlog	PSL for Verilog
pslvhdl	.pslvhdl	PSL for VHDL
C	.c	C
cpp	.cpp, .cc, .cxx	C++
as	.s	Assembly
obj	.o	Object file
so	.so .sl	Shared object
spice	scs .sp	Spice

[+] —The “+” sign adds to the association list. Without the “+” sign, associations are replaced. For example, the command “qrun -vlog.ext=.v,.vv,.vh” replaces the Verilog suffixes list. The command “qrun -vhdl.ext=+.vhdh,.xyz“ adds *.vhdh and *xyz to the VHDL list.

<tool> —The tool to associate a file type.

- +UVM_TESTNAME=<testname>

  Defines the +UVM_TESTNAME test name to be used for UVM designs.
- -uvm

  Include the UVM library. Optionally specified with -uvmhome.
- -uvmcontrol

  Control specific UVM-aware debug options.
- -uvmhome <path-to-UVM-library|UVM-version-string>

  Specifies UVM library to use with -uvm. . You can specify a path to the library or a version number.
- -uvmexthome <QUESTA_HOME/questasim/verilog_src/questa_uvm_pkg-1.2>

  Compile the QuestaSIM UVM extensions into the work library.
- -verbose

  Display verbose statistics information when available.
- -version

  Prints the version of qrun.
- -visualize

  Enable the Visualizer GUI and flow.
- -warning <msgNumber>[,<msgNumber> ...]

  Report the listed messages as warnings.
- -xcheck

  Enable xcheck flow for Formal flow.

Examples

The following shows how to associate both a single and multiple options to a tool.

```
qrun \
+vlog.acc \
-vlog.options -fsmdebug +acc -end \
-makelib toplib  t2.sv -endlib \
-makelib dutlib dut.sv -endlib
```

In this example, the qrun command causes all logs from vlog to be redirected to vlog.log. This is also the same for vopt and vcom. The logs from the rest of the programs will be only printed to the terminal unless a -l argument is used to capture them.

```
qrun -vlog.log vlog.log -vcom.log vcom.log -vopt.log vopt.log ...
```

Invoke Visualizer on a SystemVerilog design.

```
qrun *.sv -gui -visualizer
```

Apply the -pedantic switch only to Verilog files:

qrun -vlog.pedantic a.sv b.vhd

Run vsim -c -suppress 12110 using the qrun command.

qrun -vsim.c -vsim.suppress 12110 ...

Apply the +cover argument to vlog only.

qrun +vlog.cover+bcs...

Compile file1.vhd and file2.vhd into the library "mylib" with argument. -93.

qrun -compile -93 -makelib mylib file1.vhd file2.vhd -endlib

Compile files specified in mylib_files.txt into library "mylib" with arguments from -f file and -93.

qrun -compile -93 -makelib mylib -f mylib_files.txt -endlib

This command specifies a custom source file extension .myvhdl for the VHDL files specified in the -makelib collection libvhdl.

qrun -makelib libvhdl -vhdl_ext +.myvhdl bar.myvhdl -endlib tb_top.vhd

Specifying aphysical library for -makelib.

qrun -makelib /usr1/myarea/gates buf.v and2.v -endlib

The -history switch with output.

qrun -history

```
1 14:32 qrun a.vhd
2 14:50 qrun b.vhd c.sv
3 16:20 qrun -mfcu -93 +acc -o opt -c a.sv b.sv c.vhd
```

Run qrun -history to show the second executed command.

qrun -history=2

```
2 14:50 qrun b.vhd c.sv
```

Re-run the second command previously executed. Using the previous example, qrun executes “qrun b.vhd c.sv”.

qrun -history=2+redo

Pass arguments and files to the 'sccom -link' command, use -sclink <options/files> argument.

qrun -sysc -g -DMYDEF a.cc b.c -top tb -sclink -scv lib1.o lib2.o

The previous command produces:

```
sccom -link -scv lib1.o lib2.o
```

To incorporate a 'foreign' module from VHDL or Verilog, use -precompile to compile the HDL first, and use the '-scgenmod <options> <design-unit name> <output file name> -end' argument:

```
qrun -precompile myvlog.sv -end -scgenmod svmodule svmodule.h -end -sysc a.cc -top tb
```

The -mixedsvvh is a list argument specifying the list of VHDL and SV files to be shared across the mixed-language boundary.Mixed design with package sharing with -mixedsvvh argument.

```
qrun -filemap -mixedsvvh <list of VHDL and SV packages> -end
```

End-User License Agreement with EDA Software Supplemental Terms