

# ISE Simulator (ISim) 13.4 Quick Reference

## Elaborating and Generating Simulation Executable

**fuse** : For given design tops, elaborates the design and generates a simulation executable

Usage: fuse [options] [[<work>.]<design top>]

Options:

-f [ --file ] arg	Read additional options from the specified file
-h [ --help ]	Print this help message
--version	Print the compiler version
--initfile arg	Use user defined simulator init file to add to or override the settings provided by the default xilinxsim.ini file
--prj arg	Specify ISim project file containing one or more entries of 'vhdl verilog <work lib> <HDL file name>'
-L [ --lib ] arg	Specify search libraries for the instantiated design units in a Verilog or Mixed language design. Use -L -lib for each search library. The format of the argument is <name>[=<dir>] where <name> is the logical name of the library and <dir> is an optional physical directory of the library
-v [ --verbose ] arg	Specify verbosity level (0 (default), 1, or 2) for printing messages
-i [ --include ] arg	Specify directories to be searched for files included using Verilog 'include. Use -il --include for each specified search directory
-d [ --define ] arg	Define Verilog macros. Use -dl --define for each Verilog macro. The format of the macro is <name>[=<val>] where <name> is name of the macro and <value> is an optional value of the macro
--rangecheck	Enable runtime value range check for VHDL
--sourcelibdir arg	Directory for Verilog source files of uncompiled modules. Use --sourcelibdir <dirname> for each source directory
--sourcelibext arg	File extension for Verilog source files of uncompiled modules. Use --sourcelibext <file extension> for each source file extension
--sourcelibfile arg	Filename of a Verilog source file which has uncompiled modules. Use --sourcelibfile <filename> for each Verilog source file
--mindelay	Compile Verilog design units with minimum delays
--typdelay	Compile Verilog design units with typical delays (default)
--maxdelay	Compile Verilog design units with maximum delays
--notimingchecks	Ignore timing check constructs in Verilog specify block(s)
--nospecify	Ignore Verilog path delays and timing checks in Verilog specify block(s)
--work arg	Specify the work library. The format of the argument is <name>[=<dir>] where <name> is the logical name of the library and <dir> is an optional physical directory of the library
-o [ --out ] arg	Specify name for generated executable (default name: x.exe)
--mt arg	Specifies the number of sub-compilation jobs which can be run in parallel. Possible values are 'on', 'off', or an integer greater than 1. (Default:on)
--timescale arg	Specify default timescale for Verilog modules( Default: 1ns/1ps )
--override_timeunit	Override timeunit for all Verilog modules, with the specified time unit in -timescale option
--override_timeprecision	Override time precision for all Verilog modules, with the specified time precision in -timescale option
--timeprecision_vhdl arg	Specify time precision for vhdl designs(Default: 1ps)
--generic_top arg	Override generic or parameter of a top level design unit with specified value( Example: -generic_top "P1=10" )
--maxdesigndepth arg	Override maximum design hierarchy depth allowed by the elaborator (Default: 5000)
--hwcosim_instance arg	<i>HWCosim - Specify the hierarchical name of the instance that will be run on FPGA (Example: /testbench/UUT)</i>
--hwcosim_clock arg	<i>HWCosim - Specify the clock port name on the instance</i>
--hwcosim_board arg	<i>HWCosim - Specify the name of the board</i>
--hwcosim_incremental arg	<i>HWCosim - Skip the implementation phase and reuse the previously created bit file. Allowed values are: 0 and 1 (Default:0)</i>

Examples:

```
fuse top1 top2 -L unisims_ver -L xilinxcorelib_ver
fuse lib1.top1 lib2.top2 -L unisims_ver -o mysim.exe
fuse top1 top2 -prj files.prj -o mysim.exe
fuse -sourcelibdir C:/mydir -sourcelibext .v -prj test.prj work.tb
fuse -generic_top WIDTH=32 -generic_top DEVICE=VIRTEX5 -timescale 1us/1fs work.tb
```

# ISE Simulator (ISim) 13.4 Quick Reference

## Launching Simulation in Batch or GUI Mode

**<fuse generated simulation executable>**: Launches simulation

Usage: <simexecutable> [options]

Options:

-f <file_name>	Take command line options from a file
<b>-gui</b>	<b>Run with interactive GUI</b>
-h	Print this help message
-log <file_name>	Specify the log file name
-maxdeltaid <number>	Specify the maximum number of delta cycles allowed without advancing of simulation time
-nolog	Suppress log file generation
-sdfnowarn	Do not emit sdf warnings
-sdfnoerror	Treat errors found in sdf file as warning
-sdfmin [ <root>=<file>	Sdf annotate <file> at instance <root> with minimum delay
-sdfmax [ <root>=<file>	Sdf annotate <file> at instance <root> with maximum delay
-sdfm [ <root>=<file>	Sdf annotate <file> at instance <root> with typical delay
-sdfmax [ <root>=<file>	Sdf annotate <file> at instance <root> with maximum delay
-sdfroot <root>	Default instance at which SDF annotation is applied
-tclbatch <file_name>	Specify the Tcl batch file for batch mode
-testplusarg <arg>	Specify plusargs to be used by \$stest\$plusargs and \$value\$plusargs
-vcdfile <vcd_file>	Specify the vcd output file
-vcdunit <unit>	Specify the vcd output unit. Default is fs
-view <wcfg_file>	Open a wave configuration file. Use it with -gui switch.
-wdb <wdb_file>	Specify the waveform database output file

Examples:

```
<sim executable> -tclbatch isim.tcl
<sim executable> -gui
<sim executable> -tclbatch isim.tcl -gui -view mywavesettings.wcfg
```

**isimgui**: Launches static waveform viewer

Usage: isimgui [options]

Options:

-h	Print this help message
-tclbatch <file_name>	Specify the Tcl batch file to customize GUI
-view <wcfg_file   wdb_file>	Opens a Waveform Configuration file (.wcfg) or a Waveform Database file (.wdb) for viewing waveform after simulation has completed. If a WCFG file is specified to this switch, the WCFG file's associated database is loaded. If -open <WDB file> is also present on command line, then the given WDB file is opened in place of the WDB file associated with the WCFG file
-open <WDB file>	Opens a WDB file without creating a wave window as if the GUI <b>File &gt; Open</b> was performed

Examples:

```
isimgui -view isim.wcfg
isimgui -view isim.wdb
isimgui -open my.wdb -view my.wcfg
```

**Launching ISim from Project Navigator**

1. Click **Project > Design Properties** and select **ISim** as Simulator
2. On the Design pane, at the top you will see a button labeled Simulation, select that to get to Simulation View. Right underneath the View, you will see a drop down to select Behavioral, Post Translate, Post Map, or Post Route simulation. Select the desired kind of simulation you want to perform
3. Then select your testbench shown under Hierarchy and you will see ISim Simulator as a process in the Processes Pane. Expand ISim Simulator and you will see a process (e.g. Simulate Behavioral Model) to launch ISim.
4. Double click on this process to launch ISim.

**Launching ISim from XPS (EDK) for simulating an embedded system design**

1. Click Edit > Preference, click Simulation, and select ISim as your Simulator under **Select Simulator** option.
2. Click **Simulation > Generate Simulation HDL Files** to generate simulation model
3. Then, click **Simulation > Launch HDL Simulator** to launch ISim

# ISE Simulator (ISim) 13.4 Quick Reference

## Re-launching Simulation

Click the Re-launch button in the ISim GUI to re-compile the design and re-launch simulation. This feature lets you make modifications in HDL to fix issues without needing to shut down the ISim GUI.

## Example: Compile and Launch Simulation

For a mixed VHDL/Verilog design consisting of files gbl.v, file1.v, file2.v, file3.v, file4.vhd, file5.vhd and testbench tb.v with gbl and tb as two tops, using Xilinx UNISIMS and IP cores, trace top level signals and run simulation for 1 us under GUI

```
vlogcomp file1.v file2.v file3.v gbl.v
vhppcomp file4.vhd file5.vhd
fuse -L unisims_ver -L xilinxcorelib_ver work.gbl work.tb -o tb.exe
tb.exe -tclbatch isim.tcl -gui
```

where the isim.tcl file has:

```
onerror {resume} ; wave add / ; run 1 us
```

## Customizing Simulation and GUI using Tcl Commands

**bp**: Sets and deletes breakpoints in your HDL source code for debugging purposes

**describe**: Displays information about the given HDL data or block object

**dump**: Displays values of variables, generics, parameters and nets in the current scope

**help**: Displays a description with usage and syntax of the specified Tcl command

**isim condition**: Executes a set of commands based on a specified condition

**isim force**: Forces or removes a value on a VHDL signal, Verilog wire, or Verilog reg

**isim get arraydisplaylength**: Displays limit on numbers of elements for array type HDL object

**isim get radix**: Gets the global radix being used

**isim get userunit**: Displays the current default unit for time values in Tcl commands

**isim ltrace**: Turns on and off line tracing

**isim ptrace**: Turns on and off tracing of execution of processes

**isim set arraydisplaylength**: Sets the limit on the number of elements for an array type HDL object

**isim set userunit**: Sets the default unit of measurement for all time values where unit is unspecified

**isim set radix**: Sets the global radix for values

**onerror**: For batch mode, controls the behavior following a failed Tcl command

**put**: Assigns a value to a specified bit, slice or all of a variable or signal

**quit**: Exits either the simulation or the software, depending on the command option

**restart**: Restarts simulation. It sets simulation time back to 0

**resume**: Process next Tcl command even if the earlier one produced error

**run**: Starts simulation

**saif**: Creates a Switching Activity Interchange format (SAIF) file for power analysis

**scope**: Navigates the design hierarchy

**sdfanno**: Back-annotates delays from an Standard Delay Format (SDF) file to the HDL design

**show**: Displays selected aspects of the design in the Simulation Console.

**step**: Executes simulation through your HDL design, line by line, to assist with debug

**test**: Compares the actual value of a net or bus with a supplied value

**vcd**: Generates simulation results in VCD format

**wcfg new**: Creates a new wave configuration: wcfg new <filename>

**wcfg open**: Opens a wave configuration from a file: wcfg open <filename>

**wcfg save**: Saves a wave configuration: wcfg save <filename>

**wcfg select**: Selects the wave configuration file to be displayed in the active window:

```
wcfg select <wave_config_name>
```

**wave log**: Logs simulation output of HDL objects to a waveform database

**wave add**: Adds simulation objects or blocks to the specified waveform configuration

**divider add**: Adds a new divider

**group add**: Adds a new group

**virtualbus add**: Adds a new virtual bus

**marker add**: Adds a new marker

## Example: Using Tcl Commands

**Create a wave configuration, add signal, divider, virtual bus, and group in desired color and radix**

```
wcfg new
set test_group_id [group add test_group]
wave add --radix hex dcm_clk_s /tb/data2 -into $test_group_id
divider add data --color blue --into $test_group_id
wave add addr1 /tb/UUT/addr2 -into $test_group_id
set vbusId [virtualbus add mybus --radix hex]
wave add sigA --into $vbusId
wave add sigB --into $vbusId
set group_id [group add test_group]
set group_id_1 [group add group_1 --into $group_id]
set group_id_2 [group add group_2 --into $group_id]
wave add clk read_ok -into $group_id_1
wave add rst write_ok -into $group_id_2
wcfg save mywave.wcfg
```

**Apply a constant or pattern force on a signal, wire or reg**

To force signal rst to 0 starting at the current simulation time:

```
isim force rst 0
```

To force signal rst to 1 starting at 10 ns from the current simulation time and cancel forcing after 50 ns from the current simulation time:

```
isim force rst 1 -time 10 ns --cancel 50 ns
```

To apply a clock to the signal clk such that clk goes to 1 at current simulation time, goes back to 0 at 20 ns later and then repeats this every 40 ns until 1 us from the current simulation time i.e. to generate a clock with 50% duty cycle and 40 ns period for a duration of 1 us:

```
isim force clk 1 --value 0 --time 20 ns --repeat 40 ns --cancel 1 us
```

To force signal data\_in to 1 at current simulation time, set data\_in to 0 at current simulation time + 50 ns, and set data\_in back to 1 at current simulation time + 75 ns and then repeat this 101 pattern every 100 ns for a duration of 5000 ns:

```
force add data_in 1 -value 0 -time 50 ns -value 1 -time 75 ns -repeat 100 ns --cancel 5000 ns
```

To remove the values on signal s, s1 and s2:

```
isim force remove s s1 s2
```

**Write toggle counts in SAIF format to file uut\_power.saif for the ports and internal nets of a design starting at instance uut**

```
saif open -scope uut -file uut_power.saif --allnets
run 1000 ns
saif close
```

**Write the VCD simulation values of the module UUT to a VCD file**

```
vcd dumpfile adder.vcd
vcd dumpvars -m /UUT
run 1000 ns
vcd dumpflush
```

## Viewing Waveform in GUI

- After launching the ISim GUI, in the Instances and Processes (upper left) panel, click on the instance containing the desired simulation object(s) to display
- In the Objects panel (to the right of Instances and Processes) select one or more objects, using Ctrl or Shift for multiple selection
- Drag the objects to the Name column of the black waveform window. Optionally, call **File >New** and select **Waveform Configuration** to create additional waveform windows and drag desired objects to the additional waveform windows
- Run the simulation by clicking the "Play" button on the toolbar
- To bring up a floating ruler for measuring time interval between two edges, drag mouse to the left or to the right while keeping left key of the mouse pressed down

## Waveform Window Shortcut Keys

**Zoom Full:** F6 or Ctrl+Mouse left press draw line up and to left

**Zoom Out:** F7 or Ctrl+Mouse left press draw line up and to right or Ctrl+Mouse wheel

**Zoom In:** F8 or Ctrl+Mouse left press draw line down and to left or Ctrl+Mouse wheel

**Zoom To Area:** Ctrl+Mouse left press draw line down and to right

**Select Previous Signal:** Up Arrow

**Select Next Signal:** Down Arrow

**Previous Transition:** Left Arrow

**Next Transition:** Right Arrow

**Scroll Up and Down:** Mouse wheel

**Scroll Left and Right:** Shift+Mouse wheel

**Go to Time 0:** Ctrl+Home

**Go To Latest Time:** Ctrl+End

**Break simulation:** Break/Pause key

**Restart simulation:** Ctrl+Shift+F5

**Continue simulation:** F5

**Step:** F11

## ISim Project (PRJ) File

The ISim project file (PRJ) is used with the vhpcomp, vlogcomp, and fuse executables to provide a list of files associated with a design. The PRJ file contains the language, library name and the design file. Create a text file with a .prj extension (for example: my\_project.prj) and enter library and source file information as follows:

```
vhdl <library_name> <VHDL file name>
verilog <library_name> { <Verilog file names> } [-d <Macro>] [-i <Include Dir>]
```

where:

- verilog/vhdl* indicates that the source is a Verilog or VHDL file
- <library\_name>* indicates the library that a particular source on the given line should be compiled. "work" is the default library
- <file\_name>* is the source file or files associated with the library. More than one Verilog source can be specified on a given line. Only one VHDL source can be specified on a given line

Use the fuse command to compile the HDL sources specified in the PRJ file, elaborate the design, and generate the simulation executable. For example:  
 fuse -prj my\_project.prj work.top work.glbl -o my\_sim.exe

## ISim Library Mapping (.ini) File

The ISim HDL compiler executables vhpcomp, vlogcomp, and fuse use the xilinxsim.ini configuration file to learn the definitions and physical locations of VHDL and Verilog logical libraries. The compilers load xilinxsim.ini from \$XILINX/vhdl/hdp/<platform> always. In addition, a user .ini file can be specified through the -initfile switch in vlogcomp, vhpcomp, and fuse or a file named xilinxsim.ini" can be created in the current working directory to add to or override the settings provided in the default .ini file \$XILINX/vhdl/hdp/<platform>/xilinxsim.ini. The xilinxsim.ini file has the following format:

```
<logical_library>=<physical_dir_path>
```

The following is an example of a user xilinxsim.ini file:

```
mylib=C:/libs/mylib
work=C:/work
```

Visit <http://www.xilinx.com/tools/isim.htm> for more details

## Parsing (compiling) HDL Files

**vhpcomp : Parses a VHDL file and saves the parsed dump on disk**

Usage: vhpcomp {<VHDL file>}

ISim vhpcomp options:

-f [ --file ] arg	Read additional options from the specified file
-h [ --help ]	Print this help message
--version	Print the compiler version
--initfile arg	Use user defined simulator init file to add to or override the settings provided by the default xilinxsim.ini file
--prj arg	Specify ISim project file containing one or more entries of 'vhdl <work lib> <VHDL file name>'
-v [ --verbose ] arg	Specify verbosity level (0 (default), 1, or 2) for printing messages
--work arg	Specify the work library. The format of the argument is <name>[=<dir>] where <name> is the logical name of the library and <dir> is an optional physical directory of the library

Examples:

```
vhpcomp -work worklib file1.vhd file2.vhd
vhpcomp -prj files.prj
vhpcomp --work work -f file_list.txt
```

**vlogcomp : Parses a Verilog file and saves the parsed dump on disk**

Usage: vlogcomp [options] {<Verilog file>}

Options:

-f [ --file ] arg	Read additional options from the specified file
-h [ --help ]	Print this help message
--version	Print the compiler version
--initfile arg	Use user defined simulator init file to add to or override the settings provided by the default xilinxsim.ini file
--prj arg	Specify ISim project file containing one or more entries of 'verilog <work lib> <Verilog file name>'
-v [ --verbose ] arg	Specify verbosity level (0 (default), 1, or 2) for printing messages.
-i [ --include ] arg	Specify directories to be searched for files included using Verilog `include. Use -il--include for each specified search directory
-d [ --define ] arg	Define Verilog macros. Use -dl--define for each Verilog macro. The format of the macro is <name>[=<val>] where <name> is name of the macro and <value> is an optional value of the macro
--sourcelibdir arg	Directory for Verilog source files of uncompiled modules. Use --sourcelibdir <dirname> for each source directory.
--sourcelibext arg	File extension for Verilog source files of uncompiled modules. Use --sourcelibext <file extension> for each source file extension
--sourcelibfile arg	Filename of a Verilog source file which has uncompiled modules Use --sourcelibfile <filename> for each Verilog source file
--mindelay	Compile Verilog design units with minimum delays
--typdelay	Compile Verilog design units with typical delays ( Default )
--maxdelay	Compile Verilog design units with maximum delays
--work arg	Specify the work library. The format of the argument is <name>[=<dir>] where <name> is the logical name of the library and <dir> is an optional physical directory of the library

Examples:

```
vlogcomp -work worklib -d WIDTH=4 file1.v file2.v
vlogcomp --work work -i C:/myincludes -d DEVICE=VIRTEX6 f3.v
vlogcomp --work work -f file_list.txt
vlogcomp -prj files.prj
vlogcomp --sourcelibdir C:/mydir --sourcelibext .v -f files.txt
```