NAME

iverilog - Icarus Verilog compiler

SYNOPSIS

iverilog [−EiSuVv] [−Bpath] [−ccmdfile|−fcmdfile] [−Dmacro[=defn]] [−Pparameter=value] [−pflag=value] [−dname] [−g1995|−g2001|−g2005|−g2005-sv|−g2009|−g2012|−g<feature>] [−Iincludedir] [−Lmoduledir] [−mmodule] [−M[mode=]file] [−Nfile] [−ooutputfilename] [−stopmodule] [−ttype] [−Tmin/typ/max] [−Wclass] [−ypath] [−lfile] sourcefile

DESCRIPTION

iverilog is a compiler that translates Verilog source code into executable programs for simulation, or other netlist formats for further processing. The currently supported targets are *vvp* for simulation, and *fpga* for synthesis. Other target types are added as code generators are implemented.

OPTIONS

iverilog accepts the following options:

-B*base* The *iverilog* program uses external programs and configuration files to preprocess and compile the Verilog source. Normally, the path used to locate these tools is built into the *iverilog* program. However, the **−B** switch allows the user to select a different set of programs. The path given is used to locate *ivlpp*, *ivl*, code generators and the VPI modules.

-c*file* **-f***file*

These flags specify an input file that contains a list of Verilog source files. This is similar to the *command file* of other Verilog simulators, in that it is a file that contains the file names instead of taking them on the command line. See **Command Files** below.

-D*macro*

Defines macro *macro* with the string '1' as its definition. This form is normally only used to trigger ifdef conditionals in the Verilog source.

-D*macro=defn*

Defines macro *macro* as *defn*.

-P*parameter=value*

Override (i.e. defparam) a parameter in a root module. This allows the user to override at compile time (defparam) a parameter in a root module instance. For example, **−Pmain.foo=2** overrides the parameter foo in the root instance main with the value 2.

-d*name* Activate a class of compiler debugging messages. The **−d** switch may be used as often as necessary to activate all the desired messages. Supported names are scopes, eval_tree, elaborate, and synth2; any other names are ignored.

-E Preprocess the Verilog source, but do not compile it. The output file is the Verilog input, but with file inclusions and macro references expanded and removed. This is useful, for example, to preprocess Verilog source for use by other compilers.

-g1995|-g2001|-g2001-noconfig|-g2005|-g2005-sv|-g2009|-g2012

Select the Verilog language *generation* to support in the compiler. This selects between *IEEE1364−1995*, *IEEE1364−2001*, *IEEE1364−2005*, *IEEE1800−2005*, *IEEE1800−2009*, or *IEEE1800−2012*. Icarus Verilog currently defaults to the *IEEE1364−2005* generation of the language. This flag is used to restrict the language to a set of keywords/features, this allows simulation of older Verilog code that may use newer keywords and for compatibility with other tools. Much of the *IEEE1800* generations functionality is not currently supported. The *IEEE1800* generations do parse all the keywords, so they can be used to verify that *IEEE1364* compliant Verilog code does not use any of the new *IEEE1800* keywords.

-gverilog-ams|-gno-verilog-ams
Enable or disable (default) support for Verilog−AMS. Very little Verilog−AMS specific functionality is currently supported.

-gassertions|-gsupported-assertions|-gno-assertions
Enable (default) or disable SystemVerilog assertions. When enabled, assertion statements are elaborated. When disabled, assertion statements are parsed but ignored. The **−gsupported-assertions** option only enables assertions that are currently supported by the compiler.

-gspecify|-gno-specify
Enable or disable (default) specify block support. When enabled, specify block code is elaborated. When disabled, specify blocks are parsed but ignored. Specify blocks are commonly not needed for RTL simulation, and in fact can hurt performance of the simulation. However, disabling specify blocks reduces accuracy of full-timing simulations.

-gstd-include|-gno-std-include
Enable (default) or disable the search of a standard installation include directory after all other explicit include directories. This standard include directory is a convenient place to install standard header files that a Verilog program may include.

-grelative-include|-gno-relative-include
Enable or disable (default) adding the local files directory to the beginning of the include file search path. This allows files to be included relative to the current file not the more common files are only found in the working directory or in the specified include file search path.

-gxtypes|-gno-xtypes
Enable (default) or disable support for extended types. Enabling extended types allows for new types that are supported by Icarus Verilog as extensions beyond the baseline Verilog. It may be necessary to disable extended types if compiling code that clashes with the few new keywords used to implement the type system.

-gio-range-error|-gno-io-range-error
The standards requires that a vectored port have matching ranges for its port declaration as well as any net/register declaration. It was common practice in the past to only specify the range for the net/register declaration and some tools still allow this. By default any mismatch is reported as a error. Using **−gno−io−range−error** will produce a warning instead of a fatal error for the case of a vectored net/register and a scalar port declaration.

-gstrict-ca-eval|-gno-strict-ca-eval
The standard requires that if any input to a continuous assignment expression changes value, the entire expression is re-evaluated. By default, parts of the expression that do not depend on the changed input value(s) are not re-evaluated. If an expression contains a call to a function that doesn't depend solely on its input values or that has side effects, the resulting behavior will differ from that required by the standard. Using **−gstrict−ca−eval** will force standard compliant behavior (with some loss in performance).

-gstrict-expr-width|-gno-strict-expr-width
Enable or disable (default) strict compliance with the standard rules for determining expression bit lengths. When disabled, the RHS of a parameter assignment is evaluated as a lossless expression, as is any expression containing an unsized constant number, and unsized constant numbers are not truncated to integer width.

-gshared-loop-index|-gno-shared-loop-index
Enable (default) or disable the exclusion of for-loop control variables from implicit event_expression lists. When enabled, if a for-loop control variable (loop index) is only used inside the for-loop statement, the compiler will not include it in an implicit event_expression list it calculates for that statement or any enclosing statement. This allows the same control variable to be used in multiple processes without risk of entering an infinite loop caused by each process triggering all other processes that use the same varaible. For strict compliance with the standards, this behaviour should be disabled.

-I*includedir*

Append directory *includedir* to list of directories searched for Verilog include files. The **–I** switch may be used many times to specify several directories to search, the directories are searched in the order they appear on the command line.

-i Ignore missing modules. Normally it is an error if a module instantiation refers to an undefined module. This option causes the compiler to skip over that instantiation. It will also stop the compiler returning an error if there are no top level modules. This allows the compiler to be used to check incomplete designs for errors.

-L*path* This flag adds a directory to the path list used to locate VPI modules. The default path includes only the install directory for the system.vpi module, but this flag can add other directories. Multiple paths are allowed, and the paths will be searched in order.

-l*file* Add the specified file to the list of source files to be compiled, but mark it as a library file. All modules contained within that file will be treated as library modules, and only elaborated if they are instantiated by other modules in the design.

-M*path* This is equivalent to **–Mall=path**. Preserved for backwards compatibility.

-M*mode=path*

Write into the file specified by path a list of files that contribute to the compilation of the design. If **mode** is **all** or **prefix**, this includes files that are included by include directives and files that are automatically loaded by library support as well as the files explicitly specified by the user. If **mode** is **include**, only files that are included by include directives are listed. If **mode** is **module**, only files that are specified by the user or that are automatically loaded by library support are listed. The output is one file name per line, with no leading or trailing space. If **mode** is **prefix**, files that are included by include directives are prefixed by "I " and other files are prefixed by "M ".

-m*module*

Add this module to the list of VPI modules to be loaded by the simulation. Many modules can be specified, and all will be loaded, in the order specified. The system module is implicit and always included (and loaded last).

If the specified name includes at least one directory character, it is assumed to be prefixed by the path to the module, otherwise the module is searched for in the paths specified by preceding **-L** options, and if not found there, in the *iverilog* base directory.

-N*path* This is used for debugging the compiler proper. Dump the final netlist form of the design to the specified file. It otherwise does not affect operation of the compiler. The dump happens after the design is elaborated and optimized.

-o *filename*

Place output in the file *filename*. If no output file name is specified, *iverilog* uses the default name **a.out**.

-p*flag=value*

Assign a value to a target specific flag. The **–p** switch may be used as often as necessary to specify all the desired flags. The flags that are used depend on the target that is selected, and are described in target specific documentation. Flags that are not used are ignored.

-S Synthesize. Normally, if the target can accept behavioral descriptions the compiler will leave processes in behavioral form. The **–S** switch causes the compiler to perform synthesis even if it is not necessary for the target. If the target type is a netlist format, the **–S** switch is unnecessary and has no effect.

-s *topmodule*

Specify the top level module to elaborate. Icarus Verilog will by default choose modules that are not instantiated in any other modules, but sometimes that is not sufficient, or instantiates too many modules. If the user specifies one or more root modules with **–s** flags, then they will be

used as root modules instead.

-T*min*|*typ*|*max*
: Use this switch to select min, typ or max times from min:typ:max expressions. Normally, the compiler will simply use the typ value from these expressions (printing a warning for the first ten it finds) but this switch will tell the compiler explicitly which value to use. This will suppress the warning that the compiler is making a choice.

-t*target* Use this switch to specify the target output format. See the **TARGETS** section below for a list of valid output formats.

-u Treat each source file as a separate compilation unit (as defined in SystemVerilog). If compiling for an *IEEE1364* generation, this will just reset all compiler directives (including macro definitions) before each new file is processed.

-v Turn on verbose messages. This will print the command lines that are executed to perform the actual compilation, along with version information from the various components, as well as the version of the product as a whole. You will notice that the command lines include a reference to a key temporary file that passes information to the compiler proper. To keep that file from being deleted at the end of the process, provide a file name of your own in the environment variable **IVERILOG_ICONFIG**.

If the selected target is *vvp*, the **−v** switch is appended to the shebang line in the compiler output file, so directly executing the compiler output file will turn on verbose messages in *vvp*. This extra verbosity can be avoided by using the *vvp* command to indirectly execute the compiler output file.

-V Print the version of the compiler, and exit.

-W*class* Turn on different classes of warnings. See the **WARNING TYPES** section below for descriptions of the different warning groups. If multiple **−W** switches are used, the warning set is the union of all the requested classes.

-y*libdir* Append the directory to the library module search path. When the compiler finds an undefined module, it looks in these directories for files with the right name.

-Y*suffix* Add suffix to the list of accepted file name suffixes used when searching a library for cells. The list defaults to the single entry *.v*.

MODULE LIBRARIES

The Icarus Verilog compiler supports module libraries as directories that contain Verilog source files. During elaboration, the compiler notices the instantiation of undefined module types. If the user specifies library search directories, the compiler will search the directory for files with the name of the missing module type. If it finds such a file, it loads it as a Verilog source file, then tries again to elaborate the module.

Library module files should contain only a single module, but this is not a requirement. Library modules may reference other modules in the library or in the main design.

TARGETS

The Icarus Verilog compiler supports a variety of targets, for different purposes, and the **−t** switch is used to select the desired target.

null The null target causes no code to be generated. It is useful for checking the syntax of the Verilog source.

vvp This is the default. The vvp target generates code for the vvp runtime. The output is a complete program that simulates the design but must be run by the **vvp** command. The -pfileline=1 option can be used to add procedural statement debugging opcodes to the generated code. These opcodes are also used to generate file and line information for procedural warning/error messages. To

enable the debug command tracing us the trace command (trace on) from the vvp interactive prompt.

fpga This is a synthesis target that supports a variety of fpga devices, mostly by EDIF format output. The Icarus Verilog fpga code generator can generate complete designs or EDIF macros that can in turn be imported into larger designs by other tools. The **fpga** target implies the synthesis **−S** flag.

vhdl This target produces a VHDL translation of the Verilog netlist. The output is a single file containing VHDL entities corresponding to the modules in the Verilog source code. Note that only a subset of the Verilog language is supported. See the wiki for more information.

WARNING TYPES

These are the types of warnings that can be selected by the **−W** switch. All the warning types (other than **all**) can also be prefixed with **no−** to turn off that warning. This is most useful after a **−Wall** argument to suppress isolated warning types.

all This enables the anachronisms, implicit, macro-replacement, portbind, select−range, timescale, and sensitivity−entire−array warning categories.

anachronisms
This enables warnings for use of features that have been deprecated or removed in the selected generation of the Verilog language.

implicit This enables warnings for creation of implicit declarations. For example, if a scalar wire X is used but not declared in the Verilog source, this will print a warning at its first use.

macro-redefinition | macro-replacement
This enables preprocessor warnings when a macro is being redefined. The first variant prints a warning any time a macro is redefined. The second variant only prints a warning if the macro text changes. Use **no-macro-redefinition** to turn off all warnings of this type.

portbind
This enables warnings for ports of module instantiations that are not connected but probably should be. Dangling input ports, for example, will generate a warning.

select-range
This enables warnings for constant out of bound selects. This includes partial or fully out of bound selects as well as a select containing a 'bx or 'bz in the index.

timescale
This enables warnings for inconsistent use of the timescale directive. It detects if some modules have no timescale, or if modules inherit timescale from another file. Both probably mean that timescales are inconsistent, and simulation timing can be confusing and dependent on compilation order.

infloop This enables warnings for always statements that may have runtime infinite loops (has paths with no or zero delay). This class of warnings is not included in **−Wall** and hence does not have a **no−** variant. A fatal error message will always be printed when the compiler can determine that there will definitely be an infinite loop (all paths have no or zero delay).

When you suspect an always statement is producing a runtime infinite loop use this flag to find the always statements that need to have their logic verified. It is expected that many of the

warnings will be false positives, since the code treats the value of all variables and signals as indeterminate.

sensitivity-entire-vector
This enables warnings for when a part select within an "always @*" statement results in the entire vector being added to the implicit sensitivity list. Although this behaviour is prescribed by the IEEE standard, it is not what might be expected and can have performance implications if the vector is large.

sensitivity-entire-array
This enables warnings for when a word select within an "always @*" statement results in the entire array being added to the implicit sensitivity list. Although this behaviour is prescribed by the IEEE standard, it is not what might be expected and can have performance implications if the array is large.

VPI MODULES

If the source file name has a **.vpi** or **.vpl** suffix, then it is taken to be a VPI module. VPI modules supplied by the user are scanned to determine the return types of any system functions they provide. This is necessary because the compiler needs this information to elaborate expressions that contain these system functions. The module path/name is passed on to the target to allow the VPI module to be automatically loaded at the start of simulation.

VPI modules may also be supplied using the **-L** and **-m** options.

SYSTEM FUNCTION TABLE FILES [deprecated]

If the source file name has a **.sft** suffix, then it is taken to be a system function table file. A system function table file is the old method used to describe to the compiler the return types for system functions. Users are encouraged to switch to the new method of simply supplying the VPI module.

The format of the table is ASCII, one function per line. Empty lines are ignored, and lines that start with the '#' character are comment lines. Each non-comment line starts with the function name, then the vpi type (i.e. vpiSysFuncReal). The following types are supported:

vpiSysFuncReal
The function returns a real/realtime value.

vpiSysFuncInt
The function returns an integer.

vpiSysFuncSized <wid> <signed|unsigned>
The function returns a vector with the given width, and is signed or unsigned according to the flag.

vpiSysFuncString
The function returns a string. This is an Icarus-specific extension, not available in the VPI standard.

COMMAND FILES

The command file allows the user to place source file names and certain command line switches into a text file instead of on a long command line. Command files can include C or C++ style comments, as well as # comments, if the # starts the line.

file name
: A simple file name or file path is taken to be the name of a Verilog source file. The path starts with the first non-white-space character. Variables are substituted in file names.

-c *cmdfile* **-f** *cmdfile*
: A **−c** or **−f** token prefixes a command file, exactly like it does on the command line. The cmdfile may be on the same line or the next non-comment line.

-l *file* **-v** *file*
: A **−l** token prefixes a library file in the command file, exactly like it does on the command line. The parameter to the **−l** flag may be on the same line or the next non-comment line. **−v** is an alias for **−l**, provided for compatibility with other simulators.

  Variables in the *file* are substituted.

-y *libdir*
: A **−y** token prefixes a library directory in the command file, exactly like it does on the command line. The parameter to the **−y** flag may be on the same line or the next non-comment line.

  Variables in the *libdir* are substituted.

+incdir+*includedir*
: The **+incdir+** token in command files gives directories to search for include files in much the same way that **−I** flags work on the command line. The difference is that multiple *+includedir* directories are valid parameters to a single **+incdir+** token, although you may also have multiple **+incdir+** lines.

  Variables in the *includedir* are substituted.

+libext+*ext*
: The **+libext** token in command files lists file extensions to try when looking for a library file. This is useful in conjunction with **−y** flags to list suffixes to try in each directory before moving on to the next library directory.

+libdir+*dir*
: This is another way to specify library directories. See the −y flag.

+libdir-nocase+*dir*
: This is like the **+libdir** statement, but file names inside the directories declared here are case insensitive. The missing module name in a lookup need not match the file name case, as long as the letters are correct. For example, "foo" matches "Foo.v" but not "bar.v".

+define+*NAME***=***value*
: The **+define+** token is the same as the **−D** option on the command line. The value part of the token is optional.

+parameter+*NAME***=***value*
: The **+parameter+** token is the same as the **−P** option on the command line.

+timescale+*value*

The **+timescale+** token is used to set the default timescale for the simulation. This is the time units and precision before any ‘timescale directive or after a ‘resetall directive. The default is 1s/1s.

+toupper-filename

This token causes file names after this in the command file to be translated to uppercase. This helps with situations where a directory has passed through a DOS machine, and in the process the file names become munged.

+tolower-filename

This is similar to the **+toupper−filename** hack described above.

+integer-width+*value*

This allows the programmer to select the width for integer variables in the Verilog source. The default is 32, the value can be any desired integer value.

+width-cap+*value*

This allows the programmer to select the width cap for unsized expressions. If the calculated width for an unsized expression exceeds this value, the compiler will issue a warning and limit the expression width to this value.

VARIABLES IN COMMAND FILES

In certain cases, iverilog supports variables in command files. These are strings of the form "$(*varname*)" or "${*varname*}", where *varname* is the name of the environment variable to read. The entire string is replaced with the contents of that variable. Variables are only substituted in contexts that explicitly support them, including file and directory strings.

Variable values come from the operating system environment, and not from preprocessor defines elsewhere in the file or the command line.

PREDEFINED MACROS

The following macros are predefined by the compiler:

__ICARUS__ = 1

This is always defined when compiling with Icarus Verilog.

__ICARUS_SYNTH__ = 1

This is defined when synthesis is enabled.

__VAMS_ENABLE__ = 1

This is defined when Verilog−AMS is enabled.

ENVIRONMENT

iverilog also accepts some environment variables that control its behavior. These can be used to make semi-permanent changes.

IVERILOG_ICONFIG=*file-name*

This sets the name used for the temporary file that passes parameters to the compiler proper, and prevents that file being deleted after the compiler has exited.

IVERILOG_VPI_MODULE_PATH=*/some/path:/some/other/path*

This adds additional components to the VPI module search path. Paths specified in this way are searched after paths specified with **-L**, but before the default search path. Multiple paths can be separated with colons (semicolons if using Windows).

EXAMPLES

These examples assume that you have a Verilog source file called hello.v in the current directory

To compile hello.v to an executable file called a.out:

```
iverilog hello.v
```

To compile hello.v to an executable file called hello:

```
iverilog −o hello hello.v
```

To compile and run explicitly using the vvp runtime:

```
iverilog −ohello.vvp −tvvp hello.v
```

AUTHOR

Steve Williams (steve@icarus.com)

SEE ALSO

vvp(1), **<http://iverilog.icarus.com/>**

Tips on using, debugging, and developing the compiler can be found at **<http://iverilog.wikia.com/>**

COPYRIGHT