NAME

iverilog - Icarus Verilog compiler

SYNOPSIS

iverilog [-ESVv] [-Bpath] [-ccmdfile] [-g1|-g2|-g3.0] [-Dmacro[=defn]] [-pflag=value] [-Iincludedir] [-mmodule] [-Mfile] [-Nfile] [-ooutputfilename] [-stopmodule] [-ttype] [-Tmin/typ/max] [-Wclass] [-ypath] 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 *xnf* 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* This flag specifies 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*.

-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.

-g1|**-g2**|**-g3.0**

Select the Verilog language *generation* to support in the compiler. This selects between *IEEE1364-1995*(1), *IEEE1364-2001*(2), or *SystemVerilog 3.0*(3.0). Normally, Icarus Verilog defaults to the latest known generation of the language. This flag is most useful to restrict the language to a set supported by tools of specific generations, for compatibility with other tools.

-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.

-M*path* Write into the file specified by path a list of files that contribute to the compilation of the design. This includes files that are included by include directives and files that are automatically loaded by library support. The output is one file name per line, with no leading or trailing space.

-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.

-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 (with a warning) 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.

-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**.

-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.

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, they 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.

xnf This is the Xilinx Netlist Format used by many tools for placing devices in FPGAs or other programmable devices. This target is obsolete, use the **fpga** target instead.

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.

WARNING TYPES

These are the types of warnings that can be selected by the **-W** switch. All the warning types (other then **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 all supported warning categories.

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.

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.

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.

unused This enables warnings for unused variables. This may detect variables that are not assigned, or variables and nets that are not used, or similar possible misuses of variables and nets.

SYSTEM FUNCTION TABLE FILES

If the source file name as a **.sft** suffix, then it is taken to be a system function table file. A System function table file is used to describe to the compiler the return types for system functions. This is necessary because the compiler needs this information to elaborate expressions that contain these system functions, but cannot run the sizetf functions since it has no run-time.

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.

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 substitued in file names.

-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 fives 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.

+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.

VARIABLES IN COMMAND FILES

In certain cases, iverilog supports variables in command files. These are strings of the form "$(*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 substitued 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.

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
```

To compile hello.v to a file in XNF-format called hello.xnf

```
iverilog -txnf -ohello.xnf hello.v
```

AUTHOR

Steve Williams (steve@icarus.com)

SEE ALSO

vvp(1), **<http://www.icarus.com/eda/verilog/>**

COPYRIGHT