


☐

I'm not robot


reCAPTCHA

Continue

Compiler directives in verilog pdf


You can't perform that action at this time.

Princess Sumaya University for Technology
Computer Organization and Architecture I

Compiler Directives

- General syntax:
`<keyword>
- `define: similar to #define in C, used to define macros
- `<macro_name> to use the macro defined by `define
- Examples:
`define WORD_SIZE 32
`define S \$stop

`define WORD_REG reg [31:0]
`WORD_REG a_32_bit_reg;



Verilog HDL

You signed in with another tab or window. Reload to refresh your session. You signed out in another tab or window. Reload to refresh your session. [Summary] [Design Structures] [Sequential Statements] [Concurrent Statements] [Types and Constants] [Declarations] [Delay, Events] [Reserved Words] [Operators] [System Tasks] [Compiler Directives]

Compiler directives begin with "`" an accent grave, not an apostrophe Some of these would be called preprocessor commands in "C" Compilers may add additional compiler directives. They may not be portable and may not invoke the same actions. These are from the Verilog 2001 Standard. [present simple and continuous pdf worksheets](#) `include file_name // include source code from another file `define macro_name macro_code // substitute macro_code for macro_name `define macro_name(par1, par2,...) macro_code // parameterized macro `undef macro_name // undefine a macro `ifdef macro_name1 // include source lines1 if macro_name1 is defined // the source lines1 `elsif macro_name2 // any number of elsif clauses, the first defined // macro_name includes the source lines3 when no prior macro_name defined // the source lines 3 `endif // end the construct `ifndef macro_name // like `ifdef except logic is reversed, // true if macro_name is undefined `timescale 1ns/1ns // units/precision for time e.g. for %t `celldefine // marks beginning of a cell `endcelldefine // marks end of a cell `default_nettype net_type // sets the default net type for implicit // net declarations, net_type is one of: // wire, tri, tri0, tri1, triand, trior, trireg, wand, wor, none `resetall // reset all directives to default state, // undefine all macros `line number "filename" level // over rides the compilers information `unconnected_drive pull0 // set unconnected inputs to 0 `unconnected_drive pull1 // set unconnected inputs to 1 `nounconnected_drive // terminates either of the above directives `default_decay_time a_time // sets all undefined trireg net decay times, // a_time is integer, real or infinite `default_trireg_strength val // default trireg net strength 0 to 250 `delay_mode distributed // sets distributed delay mode `delay_mode path // sets path delay mode `delay_mode unit // sets unit delay mode `delay_mode zero // sets zero-delay mode Other Links Go to top Go to Verilog index Verilog is different from 'c' in compilation processing. In 'c' every source file is a compilation unit and is self-contained. All macro definition are contained within it. In verilog all declarations of macros (and all declarations in system verilog global scope) are sticky. This means that macro definitions in one source file are also seen in other source files which follow the one with declarations. So, in verilog if you want to include the same file with different macro definitions, you would need to employ `define and `undef directives, for example, `define b `include "a.sv" ... `undef b `include "a.sv" However, just a note of caution. In real projects this type of inclusions is a source of many errors, incorrect compilations and debugging problems. I suggest that you avoid using it. Compiler directives control the preprocessor part of Verilog-A compilation. These directives are capable of performing various transformations on the Verilog-A code but know nothing about the Verilog-A syntax and simply make textual changes as directed. [flappy bird multiplayer apk](#) It typically involves the inclusion of the text files, substitution of strings, conditional inclusion or exclusion of code, and setting defaults. The scope of a compiler directive is independent of module definitions and extends from the point where the directive occurs to the next compiler directive that supersedes it. This is a preview of subscription content, access via your institution. verilogams.com The `` character (referred to as the back tick, open quote, or grave accent character) introduces a language construct used to implement compiler directives. The behavior dictated by a compiler directive takes effect as soon as the compiler reads the directive. The directive remains in effect for the rest of the compilation unless a different compiler directive specifies otherwise. A compiler directive in one file can therefore control compilation behavior in multiple description files. Verilog-AMS generally support the following compiler directives. `default_discipline `default_transition `define `else `endif `ifdef `ifndef `include `resetall `timescale `undef This is only a partial list the directives that are generally available. Defines (`define) give a name to a collection of characters. Once defined, that name can be used in lieu of the characters. The name is then referred to as a macro. Any valid identifier, including keywords already in use, can be used as a name. Once defined, the macro is referenced using its name preceded by a tick. Undefines (`undef) remove the macro. Example: `define size 8 electrical {0: size-1} out; Includes (`include) are replaced by the contents of a file. It takes the filename as an argument, which can either be specified with a relative or absolute path to the file. Included files may include other files, etc.

SECAB Institute of Engineering & Technology
Department of Electronics & Communication Engineering
Accredited by NBA, New Delhi



Verilog HDL
Subject Code: 18EC56



System Tasks & Compiler Directives

Prof. Vishwanath Shidlingappanavar

Example: `include "disciplines.vams" Sections of code can be conditionally ignored using the `ifdef and `ifndef directives. It takes a macro name as an argument. [67664293642.pdf](#) With `ifdef the text that follows is ignored up to a matching `else or `endif if the argument is undefined and accepted otherwise. If `else is used, then the text between it and the matching `endif is ignored if the argument is defined, and accepted otherwise. This logic is inverted for the `ifndef directive. Verilog-AMS supports a predefined macro to allow modules to be written that work with both IEEE 1364-1995 Verilog HDL and Verilog-AMS. The predefined macro is called `VAMS_ENABLE . . . Example: `ifdef VAMS_ENABLE parameter integer del = 1 from [1:100]; `else parameter del = 1; `endif When the `resetall compiler directive is encountered during compilation, all compiler directives are set to their default values. This is useful for ensuring that only those directives that are desired when compiling a particular source file are active. To do so, place `resetall at the beginning of each source text file, followed immediately by the directives desired in the file. The `timescale compiler directive defines the time unit and the time precision for the modules that follow it. The time unit and time precision is specified using either 1, 10, or 100 followed by a measurement unit of either s, ms, us, ns, ps, or fs, which represents seconds, milliseconds, microseconds, nanoseconds, picoseconds, or femtoseconds. Example: The first value given specifies the units of time used in the file and the second specifies the precision of time. The values affect the way delays are specified and the return value from the \$realtime function. Both are rounded to the time precision and given in multiples of the time unit. Thus, with the specification given in the example above, #55.79 corresponds to a delay of 558ns (55.79 × 10ns rounded to the nearest 1 ns). Note Verilog-AMS allows numbers to be specified with the SI scale factors and so with Verilog-AMS files it is generally preferable to set the time unit to be Is and then specify the delay directly. For example: Then in the example above the delay would be specified as #357.9n rather than 55.79, which is easier to read and less error prone. © Copyright 2015-2023, Designer's Guide Consulting, Inc.. Last updated on May 27, 2023. [personal fitness merit badge pamphlet 2015 pdf](#) Built with Sphinx using a theme provided by Read the Docs. Compiler directives are common in any programming language which is based on compilers. These directives, as the name suggests, direct how the compiler will compile the code. In Verilog, there are various compiler directives to set the timescale of simulation, control the compiler flow. [pac_accessibility_checker.pdf](#) Compiler directives start with the ` (backquote) symbol. Compiler Directives, once declared, stays effective until another directive overrides the directive. Thus, if a directive is declared in one file, it is effective in other files also. Different directives available in Verilog are: `define `include `ifdef `ifndef `elsif `else `timescale `undef `resetall `default_nettypeLet us see different directives in detail.Include directiveInclude directive is used when a module defined in a file needs to be included in another file. This compiler directive will copy all the codes written in the mentioned file and include them in the present file during compile time, making all the code from another file accessible in the file. Syntax: `include "file_name" `defineThis directive is used to declare a Macro or to define a custom data type. Macros are code that can be used to perform some tasks. It is different from function or task as it can be defined outside the modules and thus be used globally. Also, macros do not have any construct like that of function and task. Syntax: `define name codeExampleThis example shows how a define directive can be used to define a custom data type. A nibble is a 4-bit data which is defined using `define directive.Copycopy code to clipboard`define nibble reg[3:0] module define_demo; `nibble a; initial begin a = 4'b1010; \$display("a = %b", a); end endmodule# a = 1010This directive is used to remove any defined macros.ExampleThis code will show an error as nibble is undefined using `undef directive. Thus, nibble cannot be used after the highlighted line.Copycopy code to clipboardmodule undefine_demo; `undef nibble `nibble a; initial begin a = 4'b1010; \$display("a = %b", a); end endmoduleThis directive is like an if-else statement but is evaluated during the compile time. If a macro has been defined, it will compile the statements present after the directive. `ifdef directive is always followed by an `endif directive which marks the end of condition code. Thus, any code written inside `ifdef and `endif directive will be compiled only when a particular macro is defined. Syntax: `ifdef macro_name `endifThe macros can either be defined using `define directive or be passed as a parameter with the compile command using the -define option.`ifndefThis directive is just the opposite of the `ifdef directive. This directive will compile the underlying code only when the macro is not defined. Generally, this is used when we want to compile some code only once. So, if a macro is not defined, it can compile some code and then define the exact macros. Now the same code will not be recompiled. Syntax: `ifndef macro_name `endif`elsifThis directive is used with the `ifdef or `ifndef directive to give added options, just as in the case of if-else-if statements.`elseThis directive is used to define a default case, i.e., if none of the directives evaluates to true, then the statement present in this directive is compiled. [jebefakoxeribova.pdf](#) It is not necessary to include an `else directive with `ifdef or `ifndef directives.`timescaleThis directive is used to define the time scale of the simulation. Choosing a correct timescale is very crucial for a simulation. The time scale is divided into two parts: time unit and time precision. Time unit maps one simulation unit to a real time unit. For example, if the time unit is selected as 1us, then #1 will mean a delay of 1us. Time precision shows the precision of the time scale. It can be equal to or less than the time unit specified. For example, if time precision is selected to be 100ns, then #1.2 would mean a time delay of 1.2us. If precision is also set to 1us, then #1.2 will mean a time delay of 1us. As there is the precision is also in us.ExampleIn the below example, it must be noted that the \$realtime function's output is related to the time precision we provide. Precision in the example is 100ns, and as 1us = 10 * 100ns, thus the output of the \$realtime function is 10.Copycopy code to clipboard`timescale 1us/100ns module timescale_demo; reg a = 1'b0; initial begin #1.4 a = 1'b1; \$display("Simulation at %0t x 100ns", \$realtime); #1.42 a = 1'b0; \$display("Simulation at %0t x 100ns", \$realtime); #1.46 a = 1'b1; \$display("Simulation at %0t x 100ns", \$realtime); end endmodule# Simulation at 14 x 100ns # Simulation at 28 x 100ns # Simulation at 43 x 100nsShare with others [qsha_guidelines_for_needle_stick_injury](#)