↖ Topic Index

Emacs support for the Verilog Hardware Description Language

Description	Keystroke	Function	Note
Verilog Editing	Emacs provide any built-in mode for the **Verilog** hardware description language support natively. PEL support for Verilog is minimal To activate PEL extra Verilog support, the **pel-use-verilog** user-option must be turned on (set to **t**). • When **pel-use-verilog is** turned on the **<f11> SPC V** prefix is made available. • PEL uses the **verilog-mode** distributed with Emacs and available as verilog-mode on GNU-Elpa. • More info about this mode is available in the veripool.org website in: Introduction to Verilog-Mode On Emacs >= 30.1, when **pel-use-tree-sitter** is on (set to t), PEL automatically activates the use of the **verilog-ts-mode** external package. • When that is installed, PEL automatically install **tree-sitter-systemverilog**, the SystemVerilog tree-sitter grammar, using the local Git command and a C and C++ compiler and linker to build the language grammar dynamic library. • Under PEL, for this automatic installation to work please follow the **PEL setup tree-sitter instructions**. The **verilog-ext** external package is used when **pel-use-verilog-ext** is turned on (set to t). The **veri-kompass** external package is used when **pel-use-veri-kompass** is turned on (set to t). ⚠ Both the Verilog and the V programming languages use the **.v** file extension! • The **v-mode** external package code does not take that into account and prevents opening Verilog .v files in verilog-mode. • PEL solves the issue in the activation of the V language support. See ꝑ - V for more information.		
Last updated on:	2025-09-14		
Open this PDF file. See also: **Help/Info**	**<f11> SPC V <f1>** **<f12> <f1>**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **ꝑ - Verilog** local PDF. If the prefix argument (like **C-u** or **M--**) is used, then it opens the remote GitHub hosted raw PDF instead. If the **pel-flip-help-pdf-arg** user-option is set it's the other way around.
Customize PEL Verilog support	**<f11> SPC V <f2>** **<f12> <f2>**	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL Verilog support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
Customize Emacs Verilog support	**<f11> SPC V <f3>** **<f12> <f3>**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs Verilog support: verilog-mode, verilog-ts, verilog-ext, veri-kompass • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
Submit bug report	**C-c C-b**	(**verilog-submit-bug-report**)	Submit via mail a bug report on verilog-mode.el.
Electric Keys			
Terminate line and indent	**RET**	(**electric-verilog-terminate-line** &optional ARG)	Terminate line & indent next line. With optional ARG, remove existing end of line comments.
: and indent	**:**	(**electric-verilog-colon**)	Insert ':' and do all indentations except line indent on this line.
; and re-indent	**;**	(**electric-verilog-semi**)	Insert ';' character and reindent the line.
` and indent to 0 if CPP	**`**	(**electric-verilog-tick**)	Insert back-tick, and indent to column 0 if this is a CPP directive.
; char and indent	**C-;**	(**electric-verilog-semi-with-comment**)	Insert ';' character, reindent the line and indent for comment.
Comments	Standard comments support and comment filling, described in **Comments** are available, plus the following Verilog specific comment control commands.		
Toggle display of comments in buffer or active region See also: **Comments**	**<f11> ; ;**	(**hide/show-comments-toggle** &optional START END) Requires the **hide-comnt.el** package PEL activates it when the **pel-use-hide-comnt** user option is **t**.	Toggle hiding/showing of comments in the active region or whole buffer. • If the region is active then toggle in the region. Otherwise, in the whole buffer.
Comment marked region	**C-c C-c**	(**verilog-comment-region** START END)	Put the region into a Verilog comment. • The comments that are in this area are "deformed": '*)' becomes '!(*' and '}' becomes '!{'. • These deformed comments are returned to normal if you use **verilog-uncomment-region** to undo the commenting. • The commented area starts with '**verilog-exclude-str-start**', and ends with '**verilog-exclude-str-end**'. • But if you change these variables, **verilog-uncomment-region** won't recognize the comments.
Insert Verilog star comment	**C-c /**	(**verilog-star-comment**)	Insert Verilog star comment at point.
Uncomment commented region	**C-c C-u**	(**verilog-uncomment-region**)	Uncomment a commented area; change deformed comments back to normal. • This command does nothing if the pointer is not in a commented area. See also 'verilog-comment-region'.
Navigation	See also **Navigation**		
Move backward to beginning of function/ procedure	**C-M-a**	(**beginning-of-defun** &optional ARG)	Move backward to the beginning of a defun. • With ARG, do it that many times. Negative ARG means move forward to the ARGth following beginning of defun.
		(**verilog-beg-of-defun**)	Move backward to the beginning of the current function or procedure.
Move backward by block	**C-M-b**	(**verilog-ts-backward-sexp** &optional ARG)	Move backward across S-expressions. With 'prefix-arg', move ARG expressions.
		(**electric-verilog-backward-sexp**)	Move backward over one balanced expression.
Contextual move downwards	**C-M-d**	(**verilog-ts-nav-down-dwim**)	Contextual based search downwards. • If inside a module or interface, navigate instances forward. • Otherwise try to find nested begin. • In any other case move one defun level down.
Move forward to end of function/procedure	**C-M-e**	(**end-of-defun** &optional ARG INTERACTIVE)	Move forward to next end of defun. • With argument, do it that many times. • Negative argument -N means move back to Nth preceding end of defun.
		(**verilog-end-of-defun**)	Move forward to the end of the current function or procedure.
Move forward by block	**C-M-f**	(**verilog-ts-forward-sexp** &optional ARG)	Move forward across S-expressions. With 'prefix-arg', move ARG expressions.
		(**electric-verilog-forward-sexp**)	Move forward over one balanced expression.
Forward to matching end	**C-M-n**	(**verilog-ts-nav-next-dwim**)	Context based search next. • If in a parenthesis, go to closing parenthesis (Elisp like). • Otherwise move through relevant language constructs.
Backward to matching end	**C-M-p**	(**verilog-ts-nav-prev-dwim**)	Context based search previous. • If in a parenthesis, go to opening parenthesis (Elisp like). • Otherwise move through relevant language constructs.
Move outside blocks	**C-M-u**	(**verilog-ts-nav-up-dwim**)	Contextual based search upwards. • If inside a module or interface, navigate instances backwards. • Otherwise if in a begin/end block move to corresponding begin. • In any other case move one defun level up.
Move to definition	**C-c C-d**	(**verilog-goto-defun**)	Move to specified Verilog module/interface/task/function. • The default is a name found in the buffer around point. • If search fails, other files are checked based on 'verilog-library-flags'.

Description	Keystroke	Function	Note
Marking	`C-M-h`	(**mark-defun** &optional ARG INTERACTIVE	Put mark at end of this defun, point at beginning.
Mark function at point		• With positive ARG, mark this and that many next defuns; with negative ARG, change the direction of marking. • If the mark is active, it marks the next or previous defun(s) after the one(s) already marked.	
See also 𝕀 **Marking**		(**verilog-mark-defun**)	Mark the current Verilog function (or procedure). Puts mark at end, point at start.
Indentation Control	See also: 𝕀 **Indentation**		
Insert new-line and indent	`M-RET`	(**electric-verilog-terminate-and-indent**)	Insert a newline and indent for the next statement.
Indent expression	`C-M-q`	(**prog-indent-sexp** &optional DEFUN)	Indent the expression after point. When interactively called with prefix, indent the enclosing defun instead.
Indent paragraph or function	`M-q`	(**prog-fill-reindent-defun** &optional ARGUMENT)	Refill or reindent the paragraph or defun that contains point. (Emacs >= 30.1) If point is in a string or a comment, fill the paragraph that contains point or follows point. Otherwise, reindent the function definition that contains or follows point.
Code Completion	See also 𝕀 **Auto-Completion**		
Completion at point	`C-M-i`	(**completion-at-point**)	Display the completions on the text around point. • The completion method is determined by '**completion-at-point-functions**'
Completion help at point	`M-?`	(**completion-help-at-point**)	Display the completions on the text around point. • The completion method is determined by '**completion-at-point-functions**'
Code Modification	Note: New users may want to set '**verilog-case-fold**' to nil and '**verilog-auto-arg-sort**' to t.		
Expand AUTO statements	`C-c C-a`	(**verilog-ts-auto** &optional INJECT)	Expand AUTO statements. Look for any /*AUTO...*/ commands in the code, as used in instantiations or argument headers. Update the list of signals following the /*AUTO...*/ command.
		(**verilog-auto** &optional INJECT)	
	More information available in the docstring of **verilog-auto**. For instance it mentions the following: • Use `M-x` **verilog-delete-auto** to remove the AUTOs. • Use `M-x` **verilog-diff-auto** to see differences in AUTO expansion. Use `M-x` **verilog-inject-auto** to insert AUTOs for the first time. • Use `M-x` **verilog-faq** for a pointer to frequently asked questions.		
Line up declarations	`C-c <tab>`	(**verilog-ts-pretty-declarations**)	Line up declarations around point.
		(**verilog-pretty-declarations** &optional QUIET)	Line up declarations around point. • Be verbose about progress unless optional QUIET set.
Delete AUTO outputs	`C-c C-k`	(**verilog-ts-delete-auto**)	Delete the automatic outputs, regs, and wires created by **verilog-ts-auto**. • Use **verilog-ts-auto** to re-insert the updated AUTOs.
		(**verilog-delete-auto**)	Delete the automatic outputs, regs, and wires created by verilog-auto. • Use verilog-auto to re-insert the updated AUTOs.
Line up expressions	`C-c C-o`	(**verilog-ts-pretty-expr**)	Line up expressions around point.
Expand vector on current line	`C-c C-e`	(**verilog-expand-vector**)	Take a signal vector on the current line and expand it to multiple lines. • Useful for creating tri's and other expanded fields.
Insert file header	`C-c C-h`	(**verilog-header**)	Insert a standard Verilog file header. • See also 'verilog-sk-header' for an alternative format.
	`C-c C-p`	(**verilog-preprocess** &optional COMMAND FILENAME)	Preprocess the buffer, similar to 'compile', but put output in Verilog-Mode. • Takes optional COMMAND or defaults to 'verilog-preprocessor', and • FILENAME to find directory to run in, or defaults to 'buffer-file-name'.
Label statements	`C-c C-r`	(**verilog-label-be**)	Label matching begin ... end, fork ... join and case ... endcase statements.
	`C-c C-s`	(**verilog-auto-save-compile**)	Update automatics with M-x verilog-auto, save the buffer, and compile.
Inject AUTO in legacy code • See command docstring for more info.	`C-c C-z`	(**verilog-inject-auto**)	Examine legacy non-AUTO code and insert AUTOs in appropriate places.
	• Any always @ blocks with sensitivity lists that match computed lists will be replaced with /*AS*/ comments. • Any cells will get /*AUTOINST*/ added to the end of the pin list. Pins with have identical names will be deleted. • Argument lists will not be deleted, /*AUTOARG*/ will only be inserted to support adding new ports. You may wish to delete older ports yourself.		
	`C-c *`	(**verilog-delete-auto-star-implicit**)	Delete all .* implicit connections created by 'verilog-auto-star'. • This function will be called automatically at save unless **verilog-auto-star-save** is set, any non-templated expanded pins will be removed.
	`C-c =`	(**verilog-pretty-expr** &optional QUIET)	Line up expressions around point. • If QUIET is non-nil, do not print messages showing the progress of line-up.
	`C-c ?`	(**verilog-diff-auto**)	Expand AUTOs in a temporary buffer and indicate any change.
	• Whitespace is ignored when detecting differences, but once adifference is detected, whitespace differences may be shown. • To call this from the command line, see **verilog-batch-diff-auto**. • The action on differences is selected with **verilog-diff-function**. It defaults to 'verilog-diff-report' to report errors & run ediff in interactive mode or 'diff' in batch.		
Syntax Checking	See also: 𝕀 **SyntaxCheck**		
Convert lint warning into disabling statements	`C-c ``	(**verilog-lint-off**)	Convert a Verilog linter warning line into a disable statement.
	• For example: `pci_bfm_null.v, line 46: Unused input: pci_rst_` becomes a comment for the appropriate tool. • The first word of **compile-command** or **verilog-linte**r variables determines which product is being used. See **verilog-surelint-off** and **verilog-verilint-off**.		
Move to next error	`C-c e n`	(**verilog-ts-goto-next-error**)	Move point to next error in the parse tree.
Move to previous error	`C-c e p`	(**verilog-ts-goto-prev-error**)	Move point to previous error in the parse tree.

Text insertion (using Emacs built-in powerful skeleton system)	The following commands are skeleton insertion commands.
	• Normally the skeleton text is inserted at point, with nothing "inside". If there is a highlighted region, the skeleton text is wrapped around the region text.
	• For these commands:
	  • A prefix argument ARG says to wrap the skeleton around the next ARG words.
	  • A prefix argument of -1 says to wrap around region, even if not highlighted.
	  • A prefix argument of zero says to wrap around zero words---that is, nothing.
	  • This is a way of overriding the use of a highlighted region.		

display comment	`C-c C-t /`	(**verilog-sk-comment** &optional STR ARG)	Inserts three comment lines, making a **display comment**.
else if statement	`C-c C-t :`	(**verilog-sk-else-if** &optional STR ARG)	Insert a skeleton **else if statement**.
inout definition	`C-c C-t =`	(**verilog-sk-inout** &optional STR ARG)	Insert an **inout definition**.
if statement	`C-c C-t ?`	(**verilog-sk-if** &optional STR ARG)	Insert an **if statement**.
assign statement	`C-c C-t A`	(**verilog-sk-assign** &optional STR ARG)	Insert an **assign statement**.
definition of signal	`C-c C-t D`	(**verilog-sk-define-signal**)	Insert a **definition of signal** under point at top of module.
function definition	`C-c C-t F`	(**verilog-sk-function** &optional STR ARG)	Insert a **function definition**.
input definition	`C-c C-t I`	(**verilog-sk-input** &optional STR ARG)	Insert an **input definition**.
output definition	`C-c C-t O`	(**verilog-sk-output** &optional STR ARG)	Insert an **output definition**.
reg definition	`C-c C-t R`	(**verilog-sk-reg** &optional STR ARG)	Insert a **reg definition**.
state machine definition	`C-c C-t S`	(**verilog-sk-state-machine** &optional STR ARG)	Insert a **state machine definition**.
class definition	`C-c C-t U`	(**verilog-sk-uvm-component** &optional STR ARG)	Insert a class definition.

Description	Keystroke	Function	Note
wire definition	`C-c C-t W`	(**verilog-sk-wire** &optional STR ARG)	Insert a **wire** definition.
always block	`C-c C-t a`	(**verilog-sk-always** &optional STR ARG)	Insert **always block**. Prompt for sensitivity list.
begin end block	`C-c C-t b`	(**verilog-sk-begin** &optional STR ARG)	Insert **begin end block**. Prompt for name.
case statement	`C-c C-t c`	(**verilog-sk-case** &optional STR ARG)	Build skeleton **case statement**, prompting for the selector expression, and the case items.
for loop	`C-c C-t f`	(**verilog-sk-for** &optional STR ARG)	Insert a skeleton **for loop** statement.
generate block	`C-c C-t g`	(**verilog-sk-generate** &optional STR ARG)	Insert **generate block**.
header	`C-c C-t h`	(**verilog-sk-header**)	Insert a **descriptive header** at the top of the file. • See also '**verilog-header**' for an alternative format.
initial block	`C-c C-t i`	(**verilog-sk-initial** &optional STR ARG	Insert an **initial block.**
fork join block	`C-c C-t j`	(**verilog-sk-fork** &optional STR ARG)	Insert a **fork join block**.
module definition	`C-c C-t m`	(**verilog-sk-module** &optional STR ARG)	Insert a **module definition**.
ovm class definition	`C-c C-t o`	(**verilog-sk-ovm-class** &optional STR ARG)	Insert a **ovm class definition.**
task definition	`C-c C-t p`	(**verilog-sk-primitive** &optional STR ARG)	Insert a **task definition**.
repeat loop	`C-c C-t r`	(**verilog-sk-repeat** &optional STR ARG)	Insert a skeleton **repeat loop** statement.
specify block	`C-c C-t s`	(**verilog-sk-specify** &optional STR ARG)	Insert **specify block**.
task definition	`C-c C-t t`	(**verilog-sk-task** &optional STR ARG)	Insert a **task definition**.
uvm class definition	`C-c C-t u`	(**verilog-sk-uvm-object** &optional STR ARG)	Insert a **uvm class definition**.
while loop	`C-c C-t w`	(**verilog-sk-while** &optional STR ARG)	Insert a skeleton **while loop** statement.
casex	`C-c C-t x`	(**verilog-sk-casex** &optional STR ARG)	Build skeleton **casex** statement, prompting for the selector expression, and the case items.
casez	`C-c C-t z`	(**verilog-sk-casez** &optional STR ARG)	Build skeleton **casez** statement, prompting for the selector expression, and the case items.
SPDX licence header **See also:** 𝕀 **Inserting Text**	• `C-c i l` • `<f6> M-l` • `<f6> i M-l`	(**spdx-insert-spdx**)	Insert a SPDX license header. Prompts for licence type. Supports tab completion. 📦 Requires **spdx** activated by **pel-use-spdx** user option.

verilog-ext commands

⚠

📦 The **verilog-ext** external package is used when **pel-use-verilog-ext** is turned on (set to t).
- **verilog-ext** provides a large set of features that can be configured via the **verilog-ext-feature-list** user-option. Access it via `<f12> <f3> 3`

🚧 **verilog-ext** features are not all described here. I will add more information later.

☝ By default, **verilog-ext,** activates all features. Some features may fail and when that happens **verilog-ext** stops configuring the others.
PEL automatically executes the **verilog-ext-mode-setup** command when **pel-use-verilog-ext** is on, catches any error but display a warning message.
- This way you can still edit the Verilog file with the features that could be enabled before the verilog-ext failure.
- Properly configure the failing feature or disable it to get more features working.

Text template insertion Hydra keys

See info on yasnippet : 𝕀 **Inserting Text**

📦 With PEL, the user option **pel-use-hydra** must be set to t, to get PEL to activate the **hydra** external package

📦 Also requires **yasnippet** activated when **pel-use-yasnippet** is set to **t** or to **use-from-start**.

`C-c C-t`	(**verilog-ext-hydra/body**)	Call the body in the "**verilog-ext-hydra**" hydra.	

✱ Hydra sequence **insert comment**

• `H`	`header - prompts`	• `/*`	`Star comment`	• `q`	Quit Hydra	Note that these hydras are exited automatically from the command.	
• `hd`	`header`	• `BC`	`Block comment`	• `C-g`	Quit Hydra		

✱ Hydra keys and their corresponding invoked functions

All these commands are executed by typing the hydra header (`C-c C-t`) followed by 1, 2 or 3 character code. Nothing else; no <tab> key is required because the commands automatically expand the yasnippets.

• `aa`	`always`	• `d`	`display`	• `gen`	`generate`	• `pkg`	`package`
• `ac`	`always_comb`	• `ei`	`else if`	• `if`	`if`	• `pgm`	`program`
• `af`	`always_ff`	• `el`	`else`	• `in`	`initial`	• `pm`	`parameter`
• `al`	`always_latch`	• `en`	`enum`	• `itf`	`interface`	• `pr`	`property`
• `ai`	`assert`	• `fl`	`final`	• `ll`	`logic`	• `rp`	`repeat`
• `ap`	`assert_prop`	• `for`	`for`	• `lv`	`logic vector`	• `seq`	`sequence`
• `as`	`assign`	• `fv`	`forever`	• `lp`	`localparam`	• `st`	`struct`
• `b`	`begin`	• `fe`	`foreach`	• `ms`	`module simple`		
• `cc`	`case`	• `fj`	`fork join`	• `md`	`module params`		
• `cls`	`class`	• `fa`	`fork join_any`	• `mp`	`modport`		
• `cb`	`clocking block`	• `fn`	`fork join_none`				
• `ct`	`constraint`	• `ff`	`function`				
• `cg`	`covergroup`						

• `ta`	`task (one line)`	• `@`	`Clk posedge`	• `uc`	`UVM Component`
• `tk`	`task (port prompt)`	• `D`	`Define signal`	• `uo`	`UVM Object`
• `td`	`typedef generic`	• `FS`	`FMS Sync`	• `ut`	`UVM TypeId Create`
• `te`	`typedef enum`	• `FA`	`FSM async`	• `ui`	`UVM Info`
• `ts`	`typedef struct`	• `IS`	`Instance simple`	• `ue`	`UVM Error`
• `tu`	`typedef union`	• `IP`	`Instance params`	• `uw`	`UVM Warning`
• `un`	`union`	• `TS`	`TB from DUT`	• `ur`	`UVM Report`
• `wh`	`while`	• `BS`	`Bind (simple)`	• `ua`	`UVM Agent`
• `wd`	`while-do`	• `BP`	`Bind (params)`		

`C-c C-l`	(**verilog-ext-formatter-run**)	Run Verible code formatter.
`C-c <f5>`	(**verilog-ext-compile-project**)	Compile using :compile-cmd of '**verilog-ext-project-alist'** project. • Depending on the command, different syntax highlight will be applied. • The function will detect any of the supported compilation error parsers and will set the appropriate mode.
`C-c C-p`	(**verilog-ext-preprocess**)	Preprocess current file. • Choose among available programs and update '**verilog-preprocessor**' variable. • Supports **verilator**, **vppreproc** and **iverilog**.
`C-c C-f`	(**verilog-ext-flycheck-mode** &optional UARG)	'**flycheck-mode**' Verilog wrapper function. • If called with UARG select among available linters and enable flycheck.
`C-c C-v`	(**verilog-ext-hierarchy-current-buffer**)	Extract and display hierarchy for module of current-buffer.
• `C-<tab>` •	(**verilog-ext-hs-toggle-hiding** &optional E)	Wrapper for '**hs-toggle-hiding**' depending on current Verilog 'major-mode'. • For '**verilog-mode**' use a modified syntax table. For '**verilog-ts-mode'** use existing one. • Toggle hiding/showing of a block. See 'hs-hide-block' and 'hs-show-block'. • Argument E should be the event that triggered this action.

Emacs & Verilog — References

Document	Notes
The Verilog Hardware Description Language	• Verilog @ Wikipedia • SystemVerilog @ Wikipedia. SystemVerilog is an extension of Verilog.
The verilog-mode	• Introduction to Verilog-Mode @ veripool.org
Other useful Emacs packages for FPGA & ASIC development	• **fpga.el** - FPGA & ASIC Utils for Emacs • **wavedrom-mode** - edit and render WaveJSON files to create timing diagrams • **vunit-mode** - interface to the VUnit unit testing framework for VHDL/SystemVerilog
Verilog Development Tools	
• **Verilog pre-processors**	Articles on verilog pre-processor: • The Verilog Preprocessor: Force for `Good and `Evil, by Wilson Snyder, 2010-08-25 • Verilog Preprocessor: Force for `Good and `Evil presentation • Comparison of Verilog free and open high-level synthesis tools @ Wikipedia

• **verilator**
  • (linter, multi-thread, convert to C++ or System-C)
  • Very popular
	• **verilator @ Wikipedia**
	• **verilator home**
	• **verilator @ Github**
	  • **verilator group @ Github**

• **vppreproc** (A Perl program, see **Perl**)	• **vppreproc** • **vppreproc command line man page**
• **iverilog** (aka Icarus Verilog, written in C++)	• **iverilog @ Wikipedia** • **iverilog @ Github**
Some older Verilog mode sites	• Mac's Verilog Mode for emacs