Electric VHDL mode:
Major mode for editing VHDL code.

Usage:

- TEMPLATE INSERTION (electrification) (`SPC'): After typing
  a VHDL keyword and entering `SPC', you are prompted for
  arguments while a template is generated for that VHDL construct. Typing
  `RET' (or `C-g' in yes-no queries) at the first
  prompt aborts the current template generation. Typing `M-SPC'
  after a keyword inserts a space without calling the template generator.
  Automatic calling of the template generators (i.e. electrification) can be
  disabled (enabled) by setting the variable `vhdl-electric-mode' to nil
  (non-nil) or by typing `C-c C-e' (toggles electrification
  mode).
  Template generators can be called using the VHDL menu, the key bindings, or
  by typing the keyword (first word of menu entry not in parenthesis) and
  `SPC'. The following abbreviations can also be used:
  arch, attr, conc, conf, comp, cons, func, inst, pack, ret, sig, sub, var.

- HEADER INSERTION (`C-c C-t h'): A customized header can be inserted
  including the actual file name, user name, and current date as well as
  prompted title strings. A custom header can be defined in a separate file
  (see custom variable `vhdl-header-file').

- STUTTERING (double strike): Double striking of some keys inserts cumbersome
  VHDL syntax elements. Stuttering can be disabled by variable
  `vhdl-stutter-mode' and be toggled by typing `C-c C-s'.

```
    ''   -->  "             [   -->  (        --    -->  comment
    ;;   -->  " : "         [[  -->  [        --CR  -->  comment-out code
    ;;;  -->  " := "        ]   -->  )        ---   -->  horizontal line
    ..   -->  " => "        ]]  -->  ]        ----  -->  display comment
    ,,   -->  " <= "        aa  -->  A    -   zz  -->  Z
```

- WORD COMPLETION (`TAB'): Typing `TAB' after a (not
  completed) word looks for a word in the buffer that starts alike and
  inserts it. Re-typing `TAB' toggles through alternative word
  completions. This also works in the minibuffer (i.e. in template generator
  prompts).

  Typing `TAB' after a non-word character indents the line if at the
  beginning of a line (i.e. no preceding non-blank characters), and inserts a
  tabulator stop otherwise. `M-TAB' always inserts a tabulator
  stop.

- COMMENTS (`--', `---', `----', `--CR'):

```
    `--'       puts a single comment.
    `---'      draws a horizontal line for separating code segments.
    `----'     inserts a display comment, i.e. two horizontal lines with a
               comment in between.
    `--CR'     comments out code on that line. Re-hitting CR comments out
               following lines.
    `C-c C-c'  comments out a region if not
               commented out, uncomments out a region if already
               commented out.
```

```
  You are prompted for comments after object definitions (i.e. signals,
  variables, constants, ports) and after subprogram and process specifications
  if variable `vhdl-prompt-for-comments' is non-nil. Comments are
  automatically inserted as additional labels (e.g. after begin statements)
  and help comments if `vhdl-self-insert-comments' is non-nil.
  Inline comments (i.e. comments after a piece of code on the same line) are
  indented at least to `vhdl-comment-column'. Comments go at maximum to
  `vhdl-end-comment-column'. `RET' after a space in a comment will
  open a new comment line. Typing beyond `vhdl-end-comment-column' in a
  comment automatically opens a new comment line. `M-q'
  re-fills multi-line comments.

- INDENTATION: `TAB' indents a line if at the beginning of the line.
  The amount of indentation is specified by variable `vhdl-basic-offset'.
  `C-c TAB' always indents the current line (is bound to `TAB'
  if variable `vhdl-intelligent-tab' is nil). Indentation can be done for
  an entire region (`M-C-\') or buffer (menu). Argument and
  port lists are indented normally (nil) or relative to the opening
  parenthesis (non-nil) according to variable `vhdl-argument-list-indent'.
  If variable `vhdl-indent-tabs-mode' is nil, spaces are used instead of tabs.
  `M-x tabify' and `M-x untabify' allow to convert spaces to tabs and vice
  versa.

- ALIGNMENT: `C-c C-a' aligns port maps, signal and
  variable assignments, inline comments, some keywords, etc., on consecutive
  lines relative to each other within a defined region.
  `C-c M-C-a' only aligns inline comments (i.e. comments
  that are at the end of a line of code). Some templates are automatically
  aligned after generation if custom variable `vhdl-auto-align' is non-nil.

- KEY BINDINGS: Key bindings (`C-c ...') exist for most commands (see in menu).

- VHDL MENU: All commands can be called from the VHDL menu.

- INDEX MENU: For each VHDL source file, an index of the contained entities,
  architectures, packages, procedures, processes, etc., is created as a menu.
  Selecting a meny entry causes the cursor to jump to the corresponding
  position in the file. Controlled by variable `vhdl-index-menu'.

- SOURCE FILE MENU: A menu containing all VHDL source files in the directory
  of the current file is generated. Selecting a menu entry loads the file.
  Controlled by variable `vhdl-source-file-menu'.

- SOURCE FILE COMPILATION: The syntax of the current buffer can be analyzed
  by calling a VHDL compiler (menu, `C-c C-k'). The compiler to be
  used is defined by variable `vhdl-compiler'. Currently supported are
  `cadence', `ikos', `quickhdl', `synopsys', `vantage', `viewlogic', and
  `v-system'. Not all compilers are tested. Please contact me for
  incorporating additional VHDL compilers. An entire hierarchy of source
  files can be compiled by the `make' command (menu, `C-c M-C-k').
  This only works if an appropriate `Makefile' exists. Compiler options can
  be defined by variable `vhdl-compiler-options'.

- KEYWORD CASE: Lower and upper case for keywords, predefined types, predefined
  attributes, and predefined enumeration values is supported. If the variable
  `vhdl-upper-case-keywords' is set to non-nil, keywords can be typed in
```

lower case and are converted into upper case automatically (not for types, attributes, and enumeration values). The case of keywords, types, attributes, and enumeration values can be fixed for an entire region (menu) or buffer (`C-c C-u') according to the variables `vhdl-upper-case-{keywords,types,attributes,enum-values}'.

- HIGHLIGHTING (fontification): Keywords, predefined types, predefined attributes, and predefined enumeration values (controlled by variable `vhdl-highlight-keywords'), as well as comments, strings, and template prompts are highlighted using different colors. Unit and subprogram names as well as labels are highlighted if variable `vhdl-highlight-names' is non-nil. The default colors from `font-lock.el' are used if variable `vhdl-customize-colors' is nil. Otherwise, an optimized set of colors is taken, which uses bright colors for signals and muted colors for everything else. Variable `vhdl-customize-faces' does the same on monochrome monitors.

  Signal highlighting allows distinction between clock, reset, status/control, data, and test signals according to some signal naming convention. Their syntax is defined by variables `vhdl-{clock,reset,control,data,test}-signal-syntax'. Signal coloring is controlled by the variable `vhdl-highlight-signals'. The default signal naming convention is as follows:

  Signal attributes:

C	clock	S	control and status
R	asynchronous reset	D	data and address
I	synchronous reset	T	test

  Syntax:
  ```
      signal name  ::=  "[A-Z][a-zA-Z0-9]*x[CRISDT][a-zA-Z0-9]*"
      signal identifier -^^^^^^^^^^^^^^^^^
      delimiter --------------------------^
      above signal attributes -------------^^^^^^^^
      additional attributes -----------------------^^^^^^^^^^^^
  ```

  (`x' is used as delimiter because `_' is reserved by the VITAL standard.)
  Examples: ClkxCfast, ResetxRB, ClearxI, SelectDataxS, DataxD, ScanEnablexT.

  If all VHDL words are written in lower case (i.e. variables `vhdl-upper-case-{keywords,types,attributes,enum-values}' are set to nil), make highlighting case sensitive by setting variable `vhdl-highlight-case-sensitive' to non-nil. This way, only names fulfilling the above signal syntax including case are highlighted.

- HIDE/SHOW: The code of entire VHDL processes or blocks can be hidden using the `Hide/Show' menu or by pressing `S-mouse-2' within the code (not in XEmacs).

- PRINTING: Postscript printing with different fonts (`ps-print-color-p' is nil, default faces from `font-lock.el' used if `vhdl-customize-faces' is nil) or colors (`ps-print-color-p' is non-nil) is possible using the standard Emacs postscript printing commands. Variable `vhdl-print-two-column' defines appropriate default settings for nice landscape two-column printing. The paper format can be set by variable `ps-paper-type'.

- CUSTOMIZATION: All variables can easily be customized using the `Customize'

  menu entry. For some variables, customization only takes effect after
  re-starting Emacs. Customization can also be done globally (i.e. site-wide,
  read INSTALL file). Variables of VHDL Mode must NOT be set using the
  `vhdl-mode-hook' in the .emacs file anymore (delete them if they still are).

Maintenance:

To submit a bug report, enter `C-c C-b' within VHDL Mode.
Add a description of the problem and include a reproducible test case.

Questions and enhancement requests can be sent to .

The `vhdl-mode-announce' mailing list informs about new VHDL Mode releases.
The `vhdl-mode-victims' mailing list informs about new VHDL Mode beta releases.
You are kindly invited to participate in beta testing. Subscribe to above
mailing lists by sending an email to .

The archive with the latest version is located at
.

Bugs and Limitations:

- Index menu does not work under XEmacs (limitation of XEmacs ?!).

- Re-indenting large regions or expressions can be slow.

- Hideshow does not work under XEmacs.

- Parsing compilation error messages for Ikos and Vantage VHDL compilers
  does not work under XEmacs.

Key bindings:

key	binding
z	vhdl-stutter-mode-caps
y	vhdl-stutter-mode-caps
x	vhdl-stutter-mode-caps
w	vhdl-stutter-mode-caps
v	vhdl-stutter-mode-caps
u	vhdl-stutter-mode-caps
t	vhdl-stutter-mode-caps
s	vhdl-stutter-mode-caps
r	vhdl-stutter-mode-caps
q	vhdl-stutter-mode-caps
p	vhdl-stutter-mode-caps
o	vhdl-stutter-mode-caps
n	vhdl-stutter-mode-caps
m	vhdl-stutter-mode-caps
l	vhdl-stutter-mode-caps

```
k               vhdl-stutter-mode-caps
j               vhdl-stutter-mode-caps
i               vhdl-stutter-mode-caps
h               vhdl-stutter-mode-caps
g               vhdl-stutter-mode-caps
f               vhdl-stutter-mode-caps
e               vhdl-stutter-mode-caps
d               vhdl-stutter-mode-caps
c               vhdl-stutter-mode-caps
b               vhdl-stutter-mode-caps
a               vhdl-stutter-mode-caps
,               vhdl-stutter-mode-comma
.               vhdl-stutter-mode-period
]               vhdl-stutter-mode-close-bracket
[               vhdl-stutter-mode-open-bracket
;               vhdl-stutter-mode-semicolon
'               vhdl-stutter-mode-quote
-               vhdl-stutter-mode-dash
SPC             vhdl-outer-space
TAB             vhdl-tab
RET             vhdl-return
DEL             backward-delete-char-untabify
ESC             Prefix Command
C-c             Prefix Command

ESC TAB         tab-to-tab-stop
ESC C-q         vhdl-indent-sexp
ESC C-h         vhdl-mark-defun
ESC C-e         vhdl-end-of-defun
ESC C-a         vhdl-beginning-of-defun
ESC C-u         vhdl-backward-up-list
ESC C-b         vhdl-backward-sexp
ESC C-f         vhdl-forward-sexp
ESC e           vhdl-end-of-statement
ESC a           vhdl-beginning-of-statement

C-c C-k         vhdl-compile
C-c C-b         vhdl-submit-bug-report
C-c C-v         vhdl-version
C-c C-h         vhdl-help
C-c C-d         vhdl-kill-line
C-c C-g         goto-line
C-c C-o         vhdl-open-line
C-c -           vhdl-inline-comment
C-c C-c         vhdl-comment-uncomment-region
C-c ESC         Prefix Command
C-c C-a         vhdl-align-noindent-region
C-c TAB         vhdl-indent-line
C-c C-r         vhdl-regress-line
C-c C-x         vhdl-show-syntactic-information
C-c C-f         font-lock-fontify-buffer
C-c C-u         vhdl-fix-case-buffer
C-c C-s         vhdl-stutter-mode
C-c C-e         vhdl-electric-mode
C-c C-t         Prefix Command

C-c ESC C-k     vhdl-make
```

```
C-c ESC -       vhdl-display-comment-line
C-c ESC C-a     vhdl-align-comment-region

C-c C-t K       Prefix Command
C-c C-t w       vhdl-while-loop
C-c C-t W       vhdl-wait
C-c C-t v       vhdl-variable
C-c C-t u       vhdl-use
C-c C-t t       vhdl-type
C-c C-t S       vhdl-subtype
C-c C-t s       vhdl-signal
C-c C-t r       vhdl-return-value
C-c C-t R       vhdl-record
C-c C-t P       vhdl-process
C-c C-t p       vhdl-procedure
C-c C-t (       vhdl-paired-parens
C-c C-t k       vhdl-package
C-c C-t n       vhdl-next
C-c C-t M       vhdl-map
C-c C-t m       vhdl-modify
C-c C-t l       vhdl-loop
C-c C-t L       vhdl-library
C-c C-t i       vhdl-if
C-c C-t h       vhdl-header
C-c C-t G       vhdl-generic
C-c C-t g       vhdl-generate
C-c C-t F       vhdl-function
C-c C-t f       vhdl-for
C-c C-t x       vhdl-exit
C-c C-t e       vhdl-entity
C-c C-t E       vhdl-elsif
C-c C-t d       vhdl-disconnect
C-c C-t C       vhdl-constant
C-c C-t I       vhdl-component-instance
C-c C-t c       vhdl-case
C-c C-t b       vhdl-block
C-c C-t A       vhdl-array
C-c C-t a       vhdl-architecture
C-c C-t ESC     Prefix Command

C-c C-t K t     vhdl-package-textio
C-c C-t K s     vhdl-package-std-logic-1164
C-c C-t K n     vhdl-package-numeric-std
C-c C-t K b     vhdl-package-numeric-bit

C-c C-t ESC W   vhdl-clocked-wait
C-c C-t ESC w   vhdl-with
C-c C-t ESC S   vhdl-selected-signal-assignment
C-c C-t ESC p   vhdl-port
C-c C-t ESC e   vhdl-else
C-c C-t ESC C   Prefix Command
C-c C-t ESC s   vhdl-concurrent-signal-assignment
C-c C-t ESC c   vhdl-component
C-c C-t ESC a   vhdl-assert
C-c C-t ESC A   vhdl-alias

C-c C-t M-C s   vhdl-configuration-spec
```

```
C-c C-t M-C d  vhdl-configuration-decl
C-c C-t M-C c  vhdl-component-configuration
C-c C-t M-C b  vhdl-block-configuration
```