

# There Will Be Code or EMACS, VHDL & Templates,

*Nigel Gunton; September 2008*

Department of Design & Engineering, BIT

<b>Grouping</b>	Individual
<b>Prerequisites</b>	Some Linux exposure or enthusiasm
<b>Courses</b>	CSI, EE, DSE, any?
<b>Requirements</b>	GNU/Linux or Unix system with Alliance tools, Stamina
<b>Summary</b>	Provides an introduction to the use of VHDL language support in EMACS
<b>Duration</b>	2 hrs

## Background

This is a quickish crib sheet for understanding VHDL mode in the EMACS editor. First check that you have the correct emacs configuration file in your home directory. On the command line, (ie. in a terminal window) type

```
cd; more1 .emacs
```

Amongst the gibberish<sup>2</sup> that appears there should be the following lines. Ask your lab tutor for help if it is missing.

```
;; needed to trigger vhdl mode for non standard file suffixes
(autoload 'vhdl-mode "vhdl-mode" "VHDL Editing Mode" t)
(setq auto-mode-alist
  (append
    '(("\.vhd$" . vhdl-mode)
      ("\.vbe$" . vhdl-mode)
      ("\.vst$" . vhdl-mode)
      ("\.fsm$" . vhdl-mode)
    ) auto-mode-alist))

;; needed to stop emacs running diff mode. It thinks pat==patch
(autoload 'text-mode "text-mode" "Text Editing Mode" t)
(setq auto-mode-alist
  (append
    '(("\.pat$" . text-mode)
    ) auto-mode-alist))
```

When running in **vhdl-mode** you will trigger a template every time that you type a VHDL keyword, or use the vhdl hot-keys. These templates then expect you to enter certain key values in the **mini-buffer** (see figure 1 for the component parts of Emacs). You are **strongly** advised to study the Emacs command worksheet to be found at

<http://www.cems.uwe.ac.uk/~ngunton/vhdl/vhd.html#worksheets>

as this worksheet uses standard Emacs commands without defining them further.

The use of the templates will speed up your coding, significantly reduce the number of syntax errors and encourage good practice in programme development. You should combine this with the use of the code management support (version control) provided within Emacs via the **Tools** menu. See also the RCS worksheet found via

<http://www.cems.uwe.ac.uk/~ngunton/worksheets/rcs.pdf>

<sup>1</sup> more allows you to view a file a page at a time. spacebar for the next page, b to go back and q to quit. *rtfm* for more details.

<sup>2</sup> Actually it's emacs lisp.

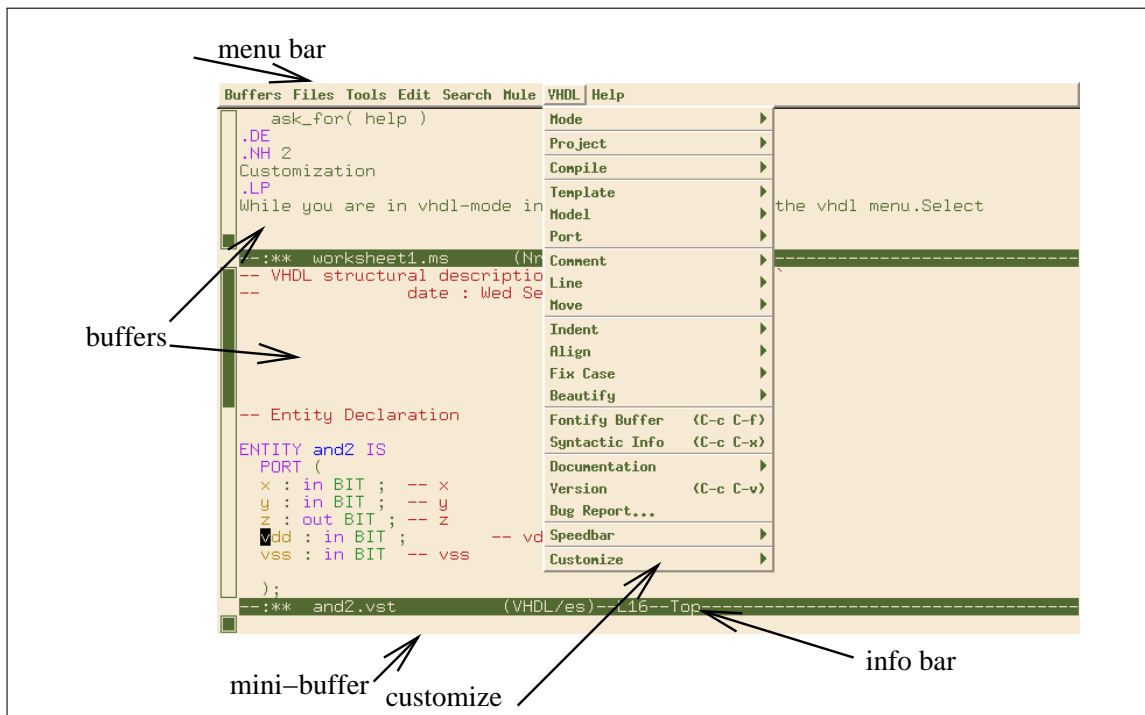


Figure 0: Emacs Components.

### Create a file:

If you haven't already done so, create a new directory<sup>3</sup> for this module, change to it and then on the command line type  
`emacs and2.vhd &`

This will launch emacs and create a new file, `and2.vhd`, in the current directory. In the main Emacs buffer type `C-c C-t en`<sup>4</sup>. This will start the template for an *entity*.



Figure 1: template start.

It will enter the keyword for you, put a marker in, in **red text**, move the cursor to the minibuffer at the bottom and wait for you to type in the **entity** name. You can cancel a template by hitting return while the field is empty in the minibuffer. Try it now, then restart the template. This time enter `and2` as the entity name and hit return.

<sup>3</sup> see [http://www.cems.uwe.ac.uk/~ngunton/worksheets/shell\\_basics.pdf](http://www.cems.uwe.ac.uk/~ngunton/worksheets/shell_basics.pdf) if you don't know how to do this.

<sup>4</sup> control-c, control-t, en := custom, template, en(tity)

```
Buffers Files Tools Edit Search Mule Minibuf Help
entity and2 is
    generic (
        <[name]>
    end and2;

--:** dummy.vhd (VHDL/es)--L4--A11-----
[ name ]: 
```

Figure 2 : *generic* subtemplate

The sub-template for *generic* is automatically started as it is an element of an *entity*. Press return as we aren't using any generics at this point. The *port* sub-template will now appear. Enter the names and values as shown in this code and in the following figures.

```
entity and2 is

    port (
        x, y : in bit;
        z    : out bit);

end and2;
```

```
Buffers Files Tools Edit Search Mule Minibuf Help
entity and2 is
    port (
        <[names]>
    end and2;

--:** dummy.vhd (VHDL/es)--L4--A11-----
[ names ]: 
```

Figure 3 : *port* sub-template, port name.

Type *x*, *y* into the minibuffer and press return. Then continue as shown in the next few diagrams.

```
Buffers Files Tools Edit Search Mule Minibuf Help
entity and2 is
    port (
        x,y : <IN | OUT | INOUT>
    end and2;

--:** dummy.vhd (VHDL/es)--L4--A11-----
IN | OUT | INOUT: in
```

Figure 4 : port sub-template, port *direction*.

```
Buffers Files Tools Edit Search Mule Minibuf Help
entity and2 is
    port (
        x,y : in <type>
    end and2;

--:** dummy.vhd (VHDL/es)--L4--A11-----
type: bit
```

Figure 5 : port sub-template, port *type*.

```
Buffers Files Tools Edit Search Mule Minibuf Help
entity and2 is
    port (
        x,y : in bit;
    end and2;
-- <[comment]>

--:** dummy.vhd (VHDL/es)--L4--A11-----
[comment]:
```

Figure 6 : port sub-template, port *comment*.

Repeat the sequence with the values for the next line of the *port* declaration and you should end up with something like *Figure 7*. If you get in a complete tangle, type C-g a few times to cancel any macros that are running, delete any text and start again. If you are really, really, stuck then ask your lab tutor.

```

entity and2 is
  port (
    x,y : in bit;
    z : out bit;
    <[names]>
  )
end and2;

```

--:\*\* dummy.vhd (VHDL/es)--L6--A11-----  
[names]:

Figure 7: completed *entity*, hit enter on empty *name* field to end.

### A Diversion

There are three ways to start a VHDL template:

Hot keys	keyword
C-c C-t en	entity
C-c C-t ar	architecture
C-c C-t si	signal
C-c C-t it	if then
C-c C-t el C-c C-t ei	else elsif

1) By using the 'hot keys' C-c C-t followed by the first 2 letters of the VHDL keyword. There are a few exceptions to this but it is true for the commonly used keywords. There are examples of both in the adjacent table.

2) By using the mouse and selecting from the menu bar VHDL → templates. A good way to learn the hot keys.

3) Typing the keyword; as soon as you press the space bar the template for that keyword will be triggered. If you don't want to run the template than just press the return/enter key to cancel the template.

Using the templates also ensures that most of the formatting is done automatically, producing easy to read code. For example templates will add the *begin*, *end*, the *then*, *end if* statements and similar for you. This saves a lot of typing.

There are many other vhd-mode shortcuts that you can learn. One of the most useful is in creating the <= and => symbols. Normally this requires use of the shift key which slows things done. Typing , , or . . quickly will generate these characters.

After the first simple exercises you should get into the habit of **always** including a header in your code. Type C-c C-t C-h or wave the mouse at the menu-bar VHDL → templates → insert header to run the template.

Happy coding.