Design Entry & Management

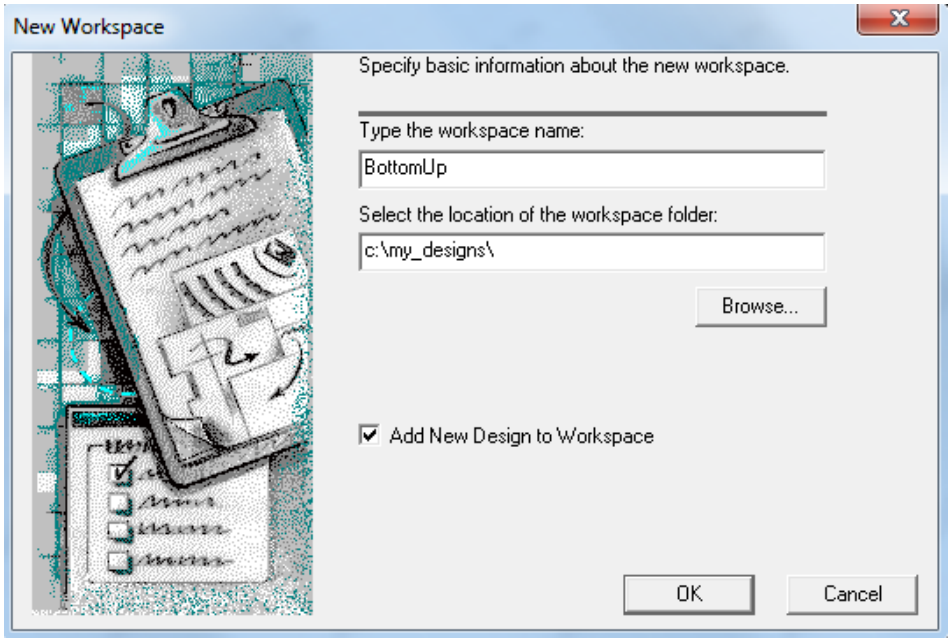
Learn how to create HDL Text Modules

Bottom-Up Design Concepts

- Start by creating a new workspace and design
  - Use the **New Source File** Wizards
  - Add existing files
  - Create an empty design
- Complete the source code
- Check syntax for errors
- Verify the functionality of the design
- Create a top-level diagram or entity

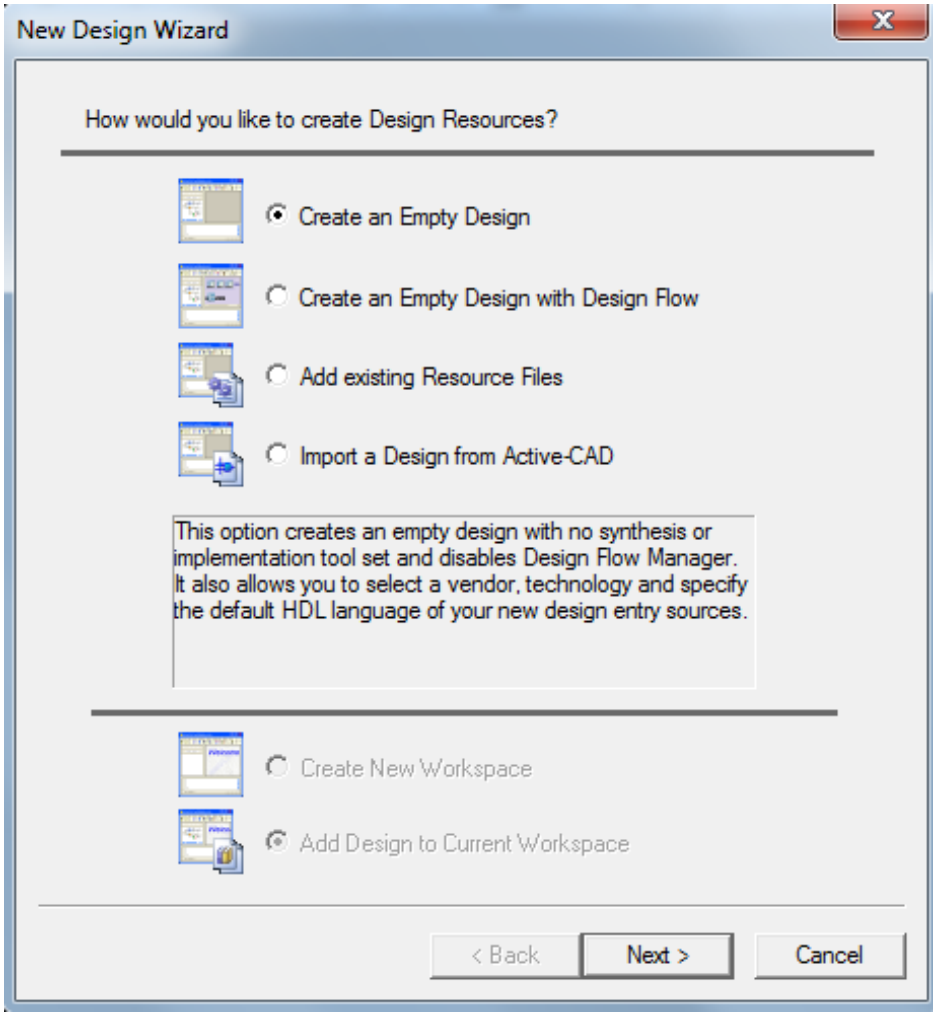
Creating Bottom-Up Design

- A *workspace* is a group of individual designs along with their resources (source files, output files with simulation results, etc.) Your workspace will allow you to add and work with several designs simultaneously.
- To create a new workspace, go to **File | New** and select **Workspace**. The **New Workspace** window will pop up and ask to type the workspace name. (For Example, BottomUp)
- If the **Add New Design to Workspace** box is checked, the **New Design Wizard** will pop up once you click OK.



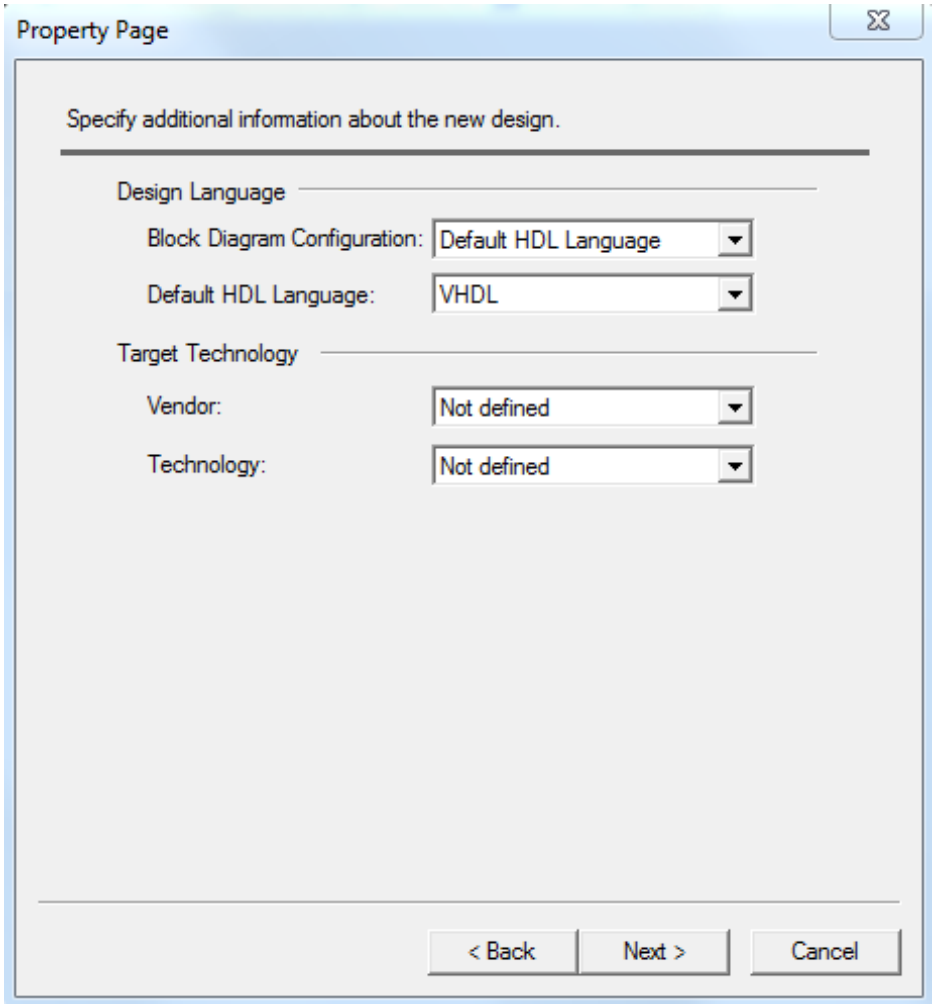
Add New Design Wizard

- The first window of the **New Design Wizard** will show you four options of design entry methods:
- Create an Empty Design
- Create an Empty Design with Design Flow
- Add existing Resource Files
- Import a Design from Active-CAD



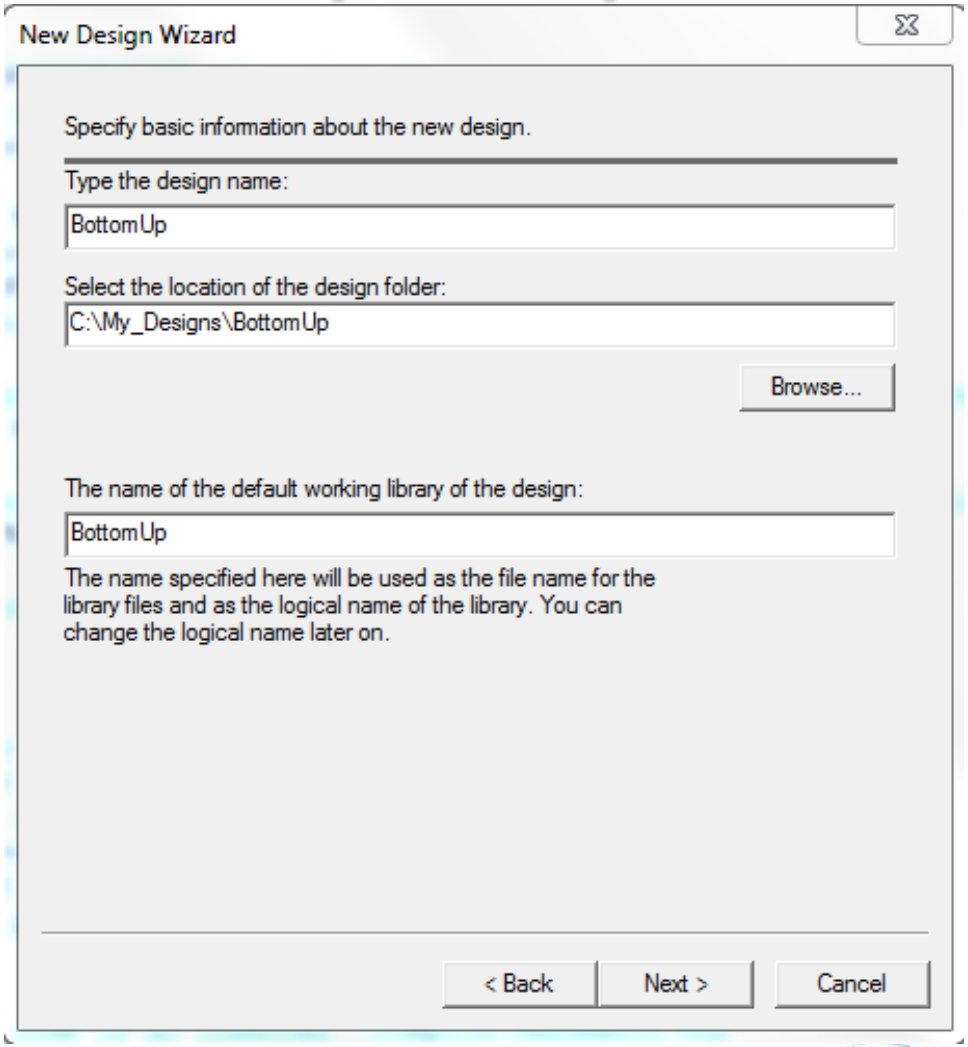
Add New Design Wizard (cont.)

- The second window is the **Property Page**. This is where you can specify additional information about your design:
- Design Language
  - Block Diagram Configuration
  - Default HDL Language
    - VHDL, Verilog
- Target Technology
  - Vendor
    - Actel, Altera, Lattice, etc.
  - Technology



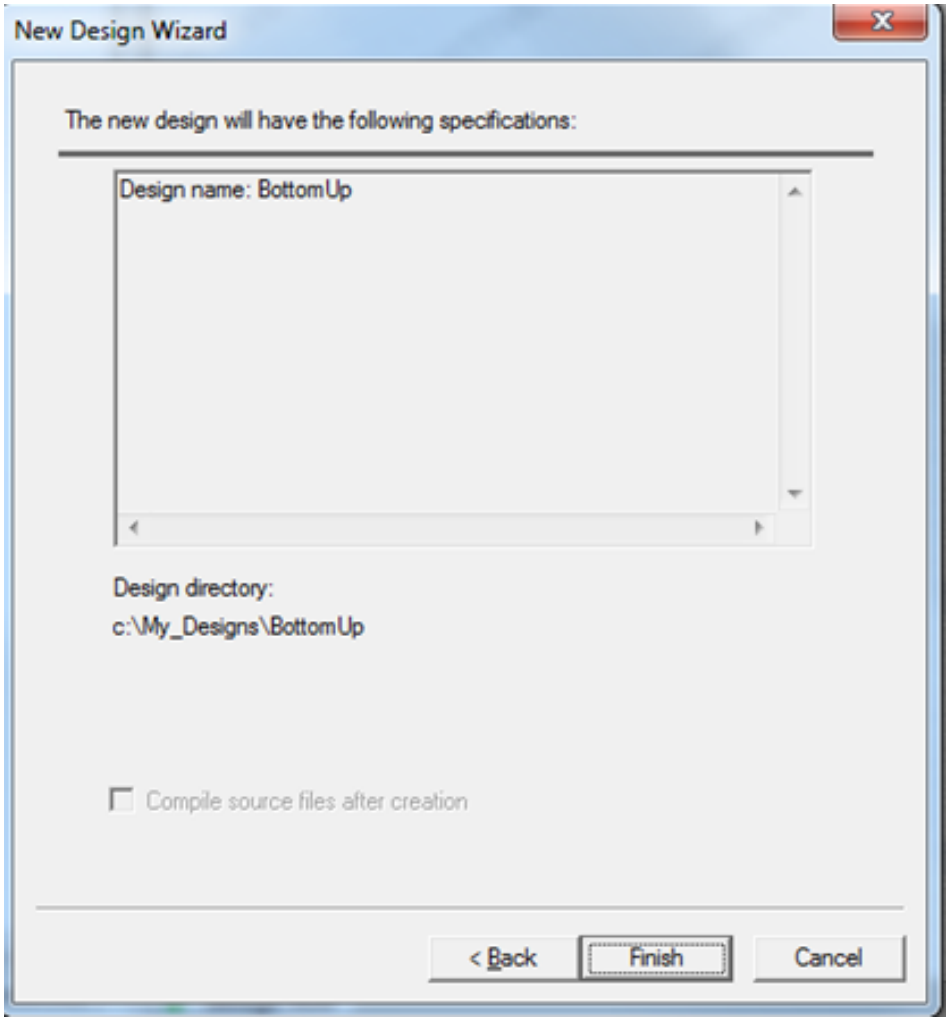
Add New Design Wizard (cont.)

- The third window is where you will be asked to specify:
- Name of your design
- Location of the design folder
- Name of the working library
  - By default, the working library name is the same as the name of your design



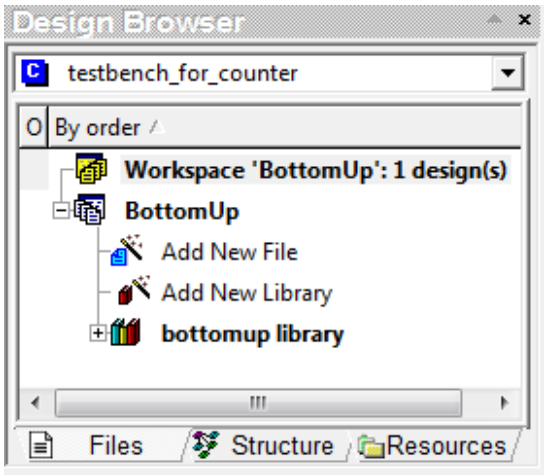
Add New Design Wizard (cont.)

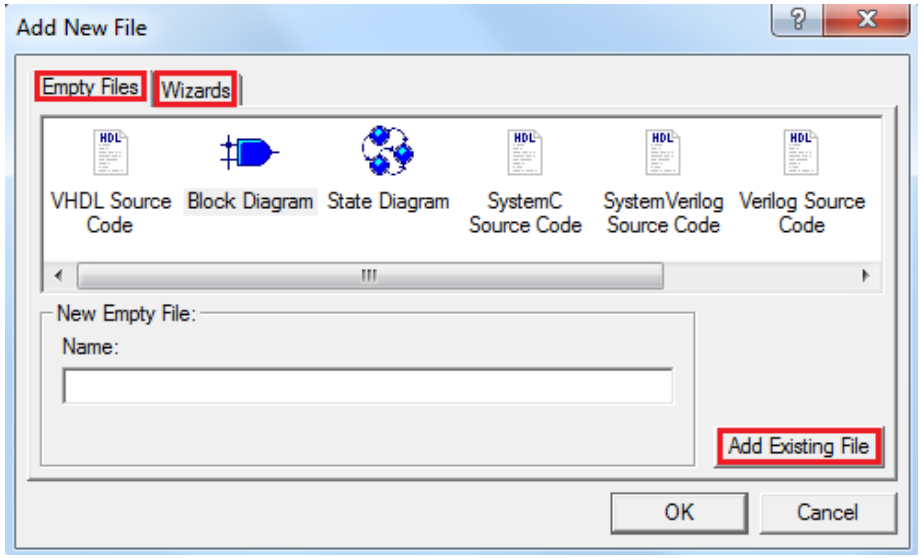
- The last window will show you a summary of all the specifications you defined in the previous windows.



Add New File: VHDL Source Code

- In your **Design Browser**, you can observe the creation of your workspace and design. To add a new or existing file to your design, double-click **Add New File**



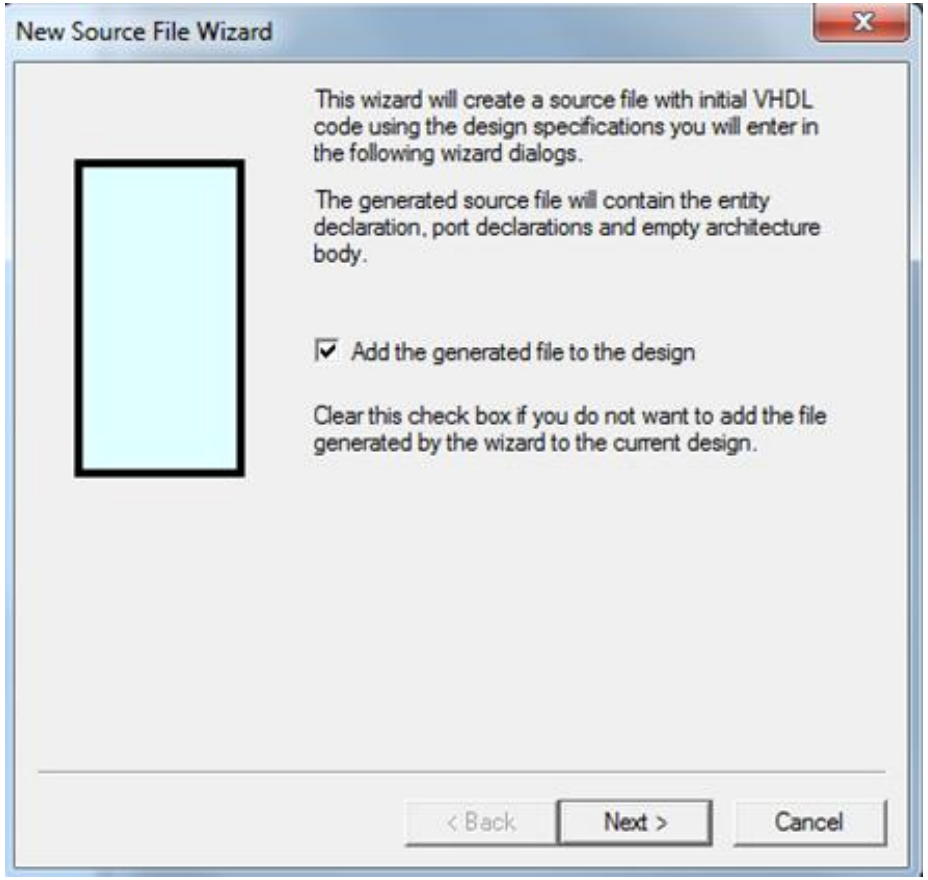


- You can create a new VHDL Source Code from scratch using the **Empty Files** tab, or with a step-by-step wizard under the **Wizards** tab.
- To add an existing file, select the **Add Existing File** button

New Source File Wizard

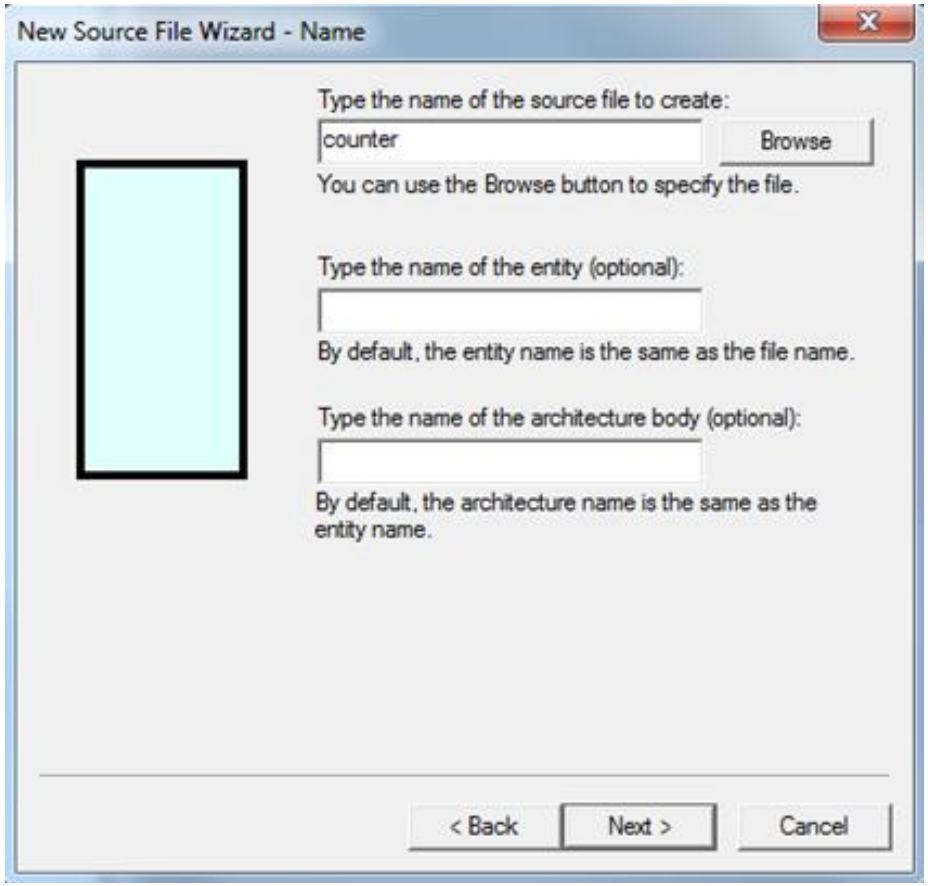
- By selecting **VHDL Source Code** under the **Wizards** tab, a **New Source File Wizard** will pop up.

- Be sure to check **Add the generated file to the design** if you want to add this to your current design.



New Source File Wizard (cont.)

- The second window of the wizard will ask you to specify:
- Name of the source file
  - For example, counter
- Name of the entity (optional)
- Name of the architecture body (optional)
  - By default, the name of the architecture is the same as the entity

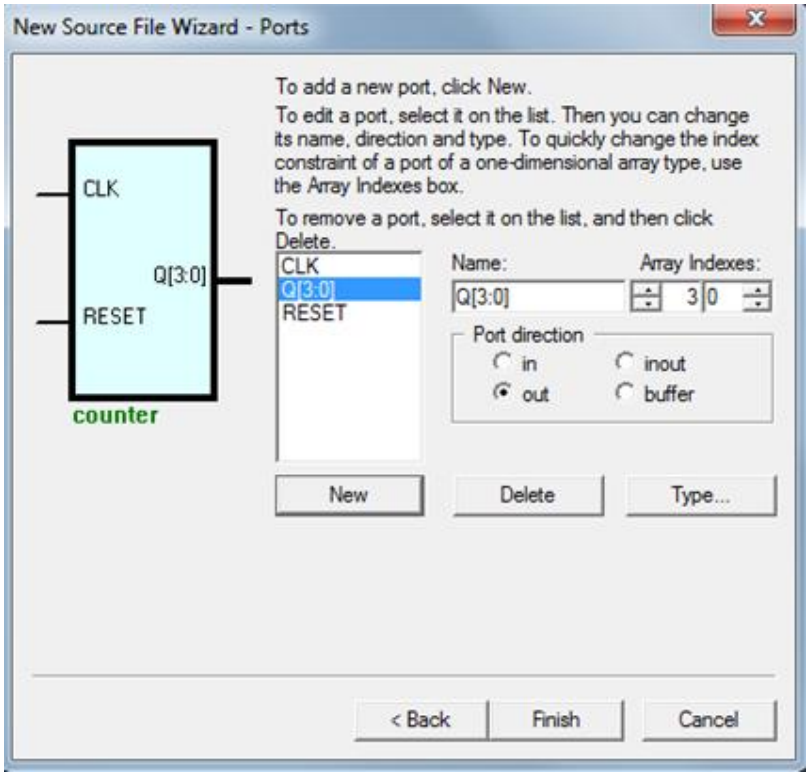


New Source File Wizard (cont.)

- The next window will ask you to define **Ports**. For each port, you will need to specify :
- Port Name
- Port direction
  - in, inout, out, buffer
- Array Index: Bus width (if needed)

For example:

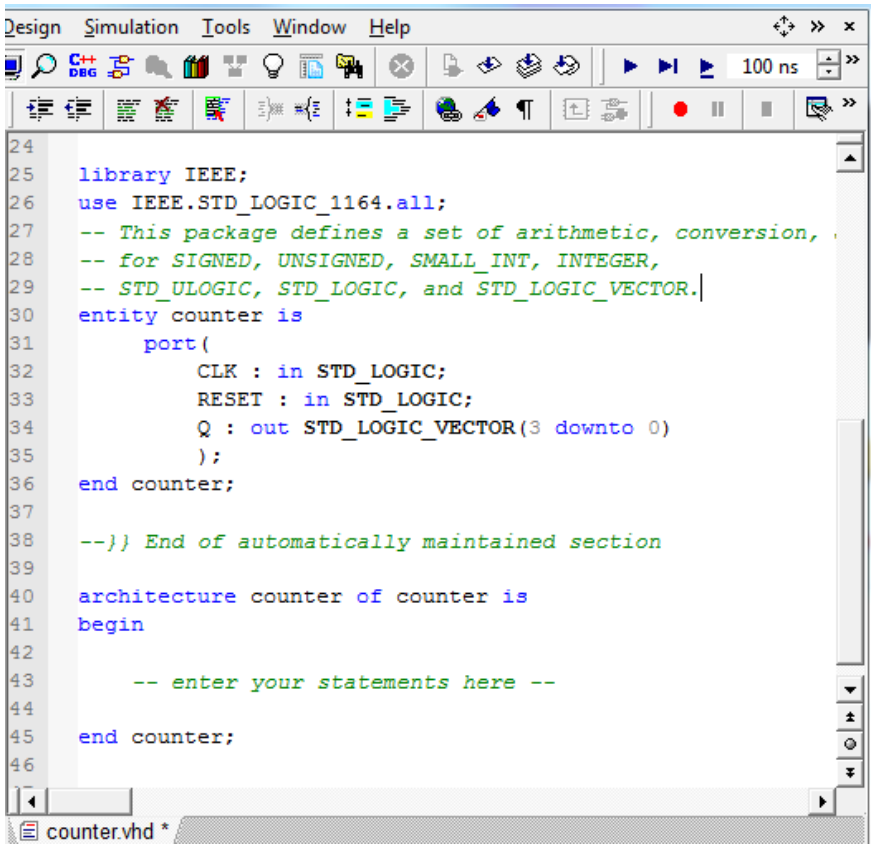
- CLK, RESET
  - Port direction: in
- Q
  - Port direction: out
  - Array Index: [3:0]



Generated VHDL Source Code

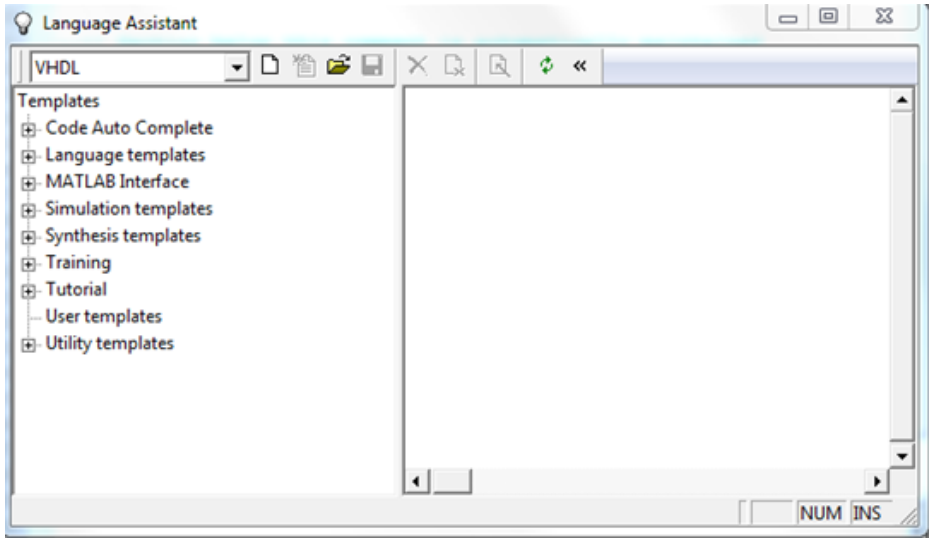
- Once you have finished the wizard, you will see your VHDL Source Code skeleton in the HDL Editor window.

- Notice that the architecture section of your code is empty. Active-HDL has built-in templates that you can drag and drop into your code using the **Language Assistant**.



Language Assistant

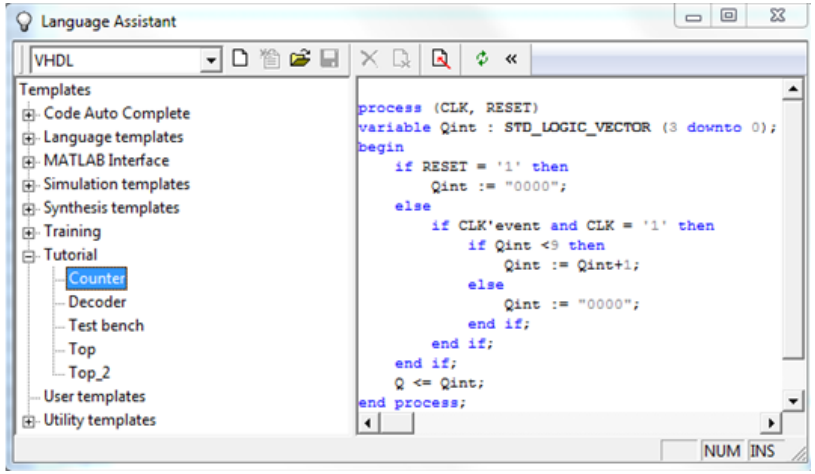
- The **Language Assistant** window contains the templates of frequently used models, user-defined models, and VHDL/Verilog constructs.

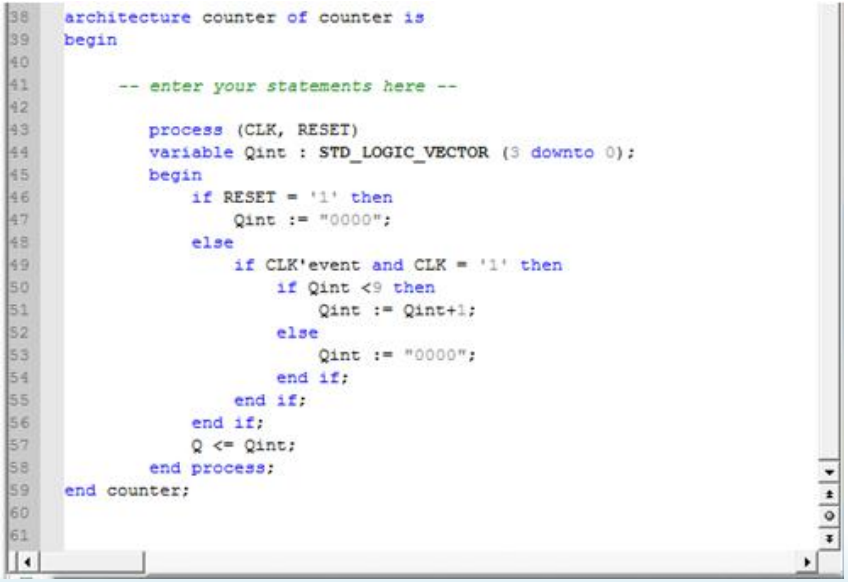


- To open the **Language Assistant**, go to **Tools | Language Assistant** or select the **Language Assistant** icon in the HDL Editor toolbar.

Language Assistant (cont.)

- To complete the architecture section of your code:
- Expand the **Tutorial** section
- Select **Counter**
- Drag and drop **Counter** to the portion of your code that states:
  - *Enter your statements here*



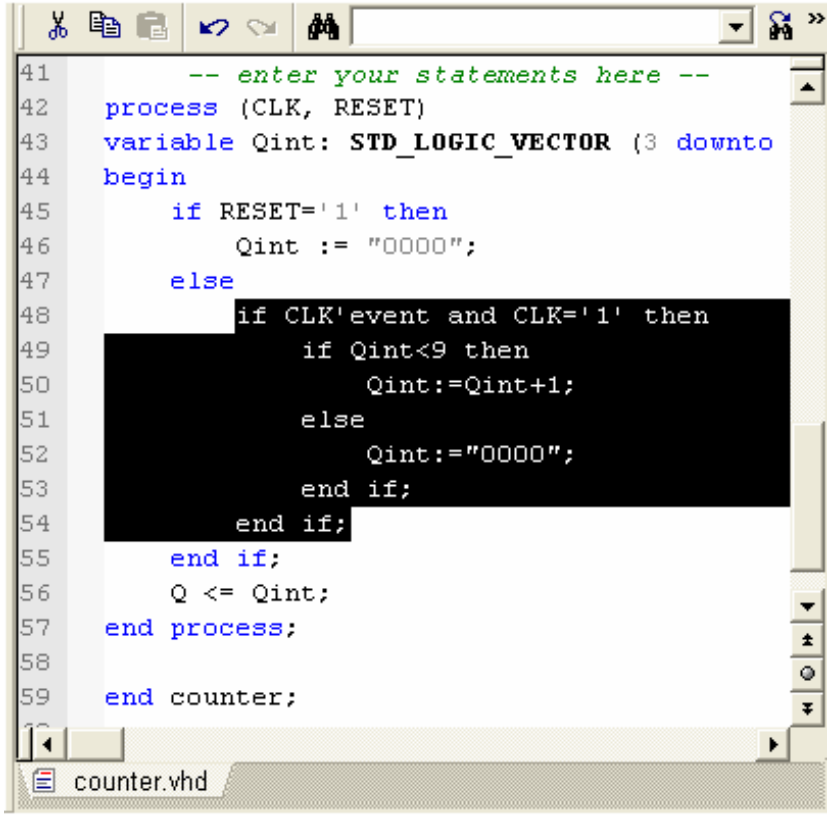


HDL Editor Features

- The HDL Editor has many features that allow the user to efficiently manage their code by:
- Keyword coloring for VHDL, Verilog/SystemVerilog, and C/C++/Handel-C
- Comment selected parts of code
- Create groups out of highlighted blocks
- Automatically creates structure for the source code
- Auto-formats the source code
- Set bookmarks throughout the code for easy navigation
- Highlights incorrect constructs after compilation
- Find and Replace given strings
- Record and Play keystrokes and macros

Marking Blocks

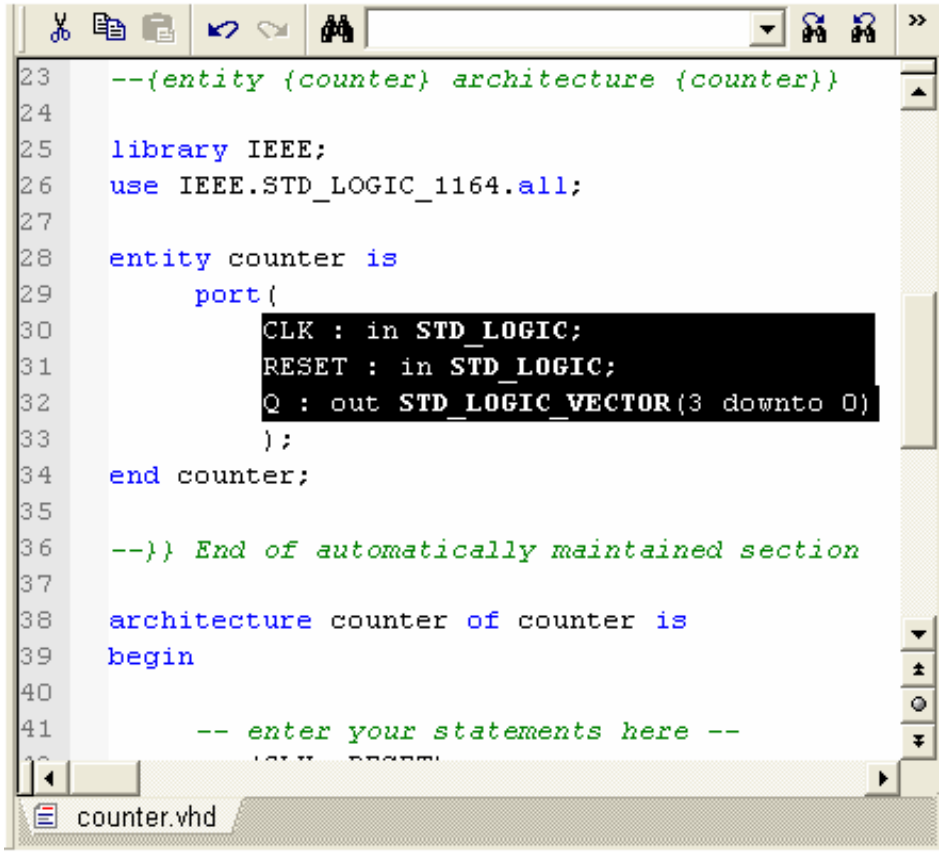
- You can select blocks of code to perform various functions. (i.e: indentation, comment, generating structure, etc.)
- There are two ways to select a block:
- Hold down the left mouse button and drag the cursor over the text to highlight the desired block.
- Hold down the **Shift** key and use the arrow keys to highlight the desired block.



Note: Whole words can be selected by pressing **Ctrl** and **Shift** together

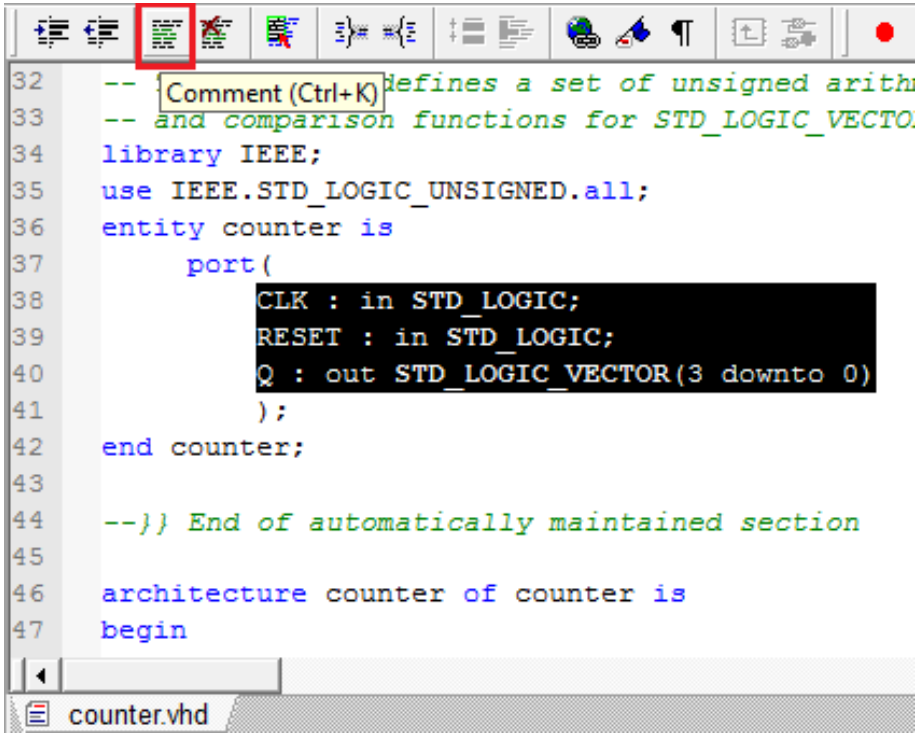
Marking Columns

- You can also select columns within a code if you wish to avoid extra white space.
- There are two ways to mark columns:
- Hold down the **Alt** key, press the left mouse button and move the cursor to highlight the desired column.
- Press **Alt+C** and use the **Shift** and **arrow** keys to highlight the desired column.



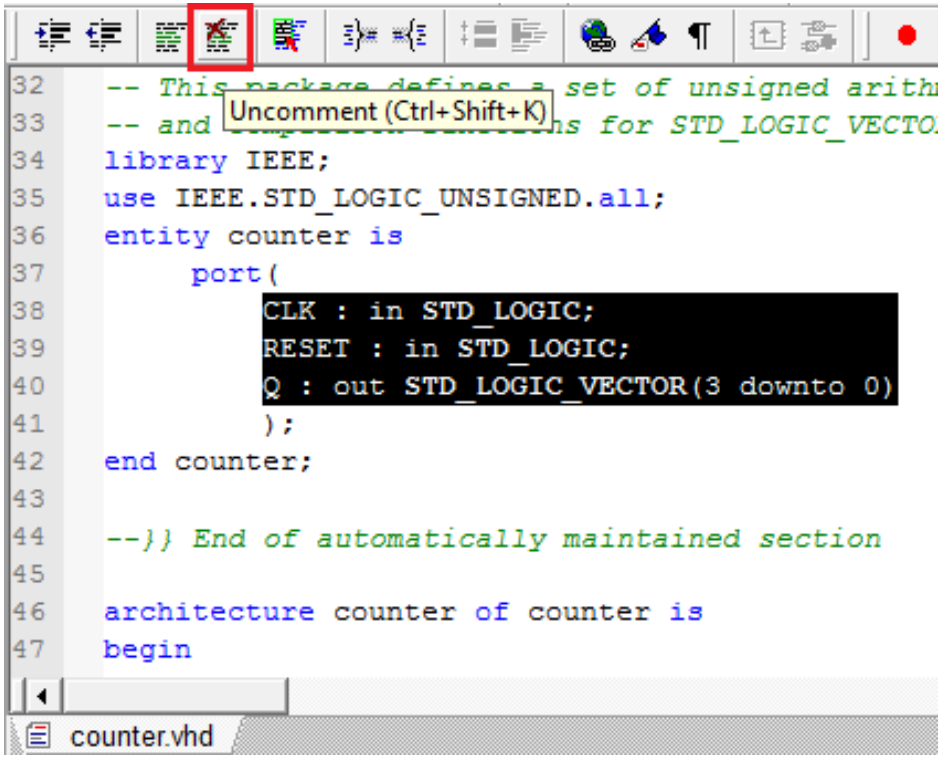
Commenting Blocks/Columns

- Sometimes you will want to comment entire blocks/columns of code either for debugging or you are not sure about the efficiency of that block.
- There are two ways to comment blocks/columns:
- Select the desired block and click the comment icon in the HDE toolbar.
- Select the desired block and press **Ctrl+K**



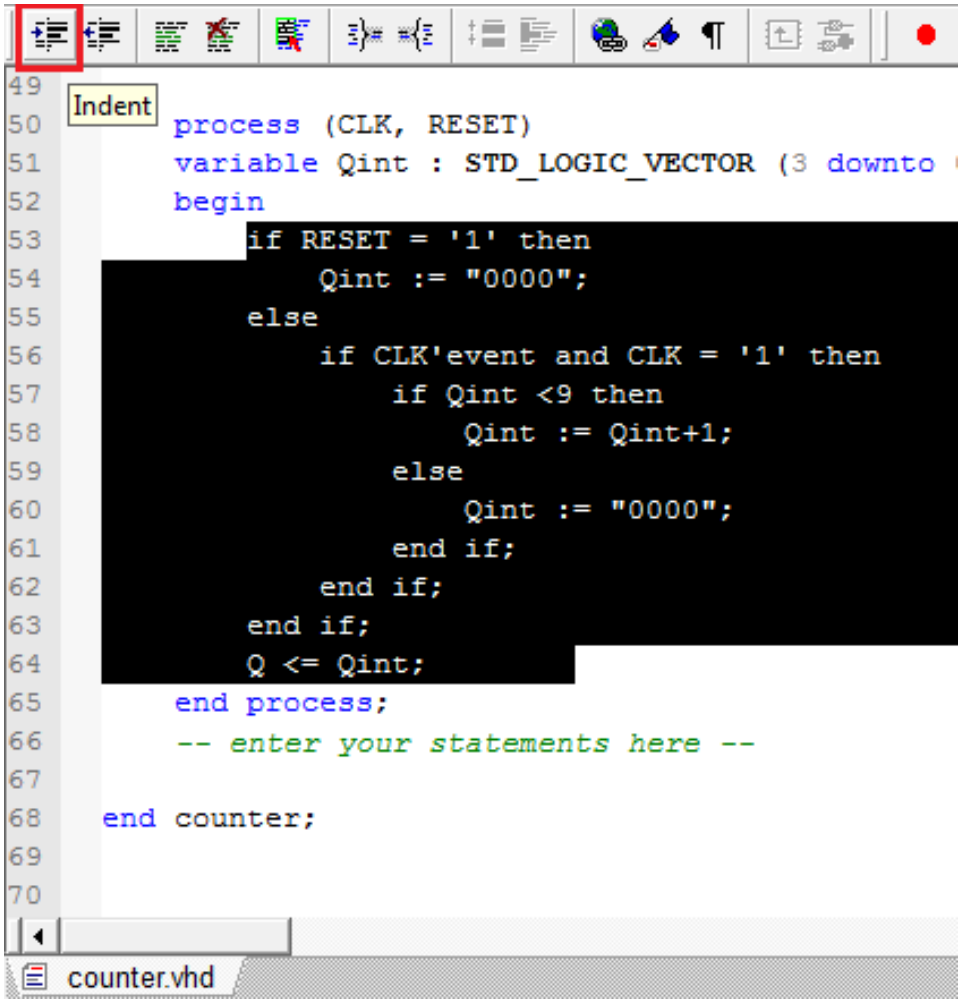
Uncommenting Blocks/Columns

- There are two ways to uncomment blocks/columns:
- Select the desired block/column, and use the **Uncomment** icon in the HDE toolbar.
- Select the desired block/column, and press **Ctrl+Shift+K**



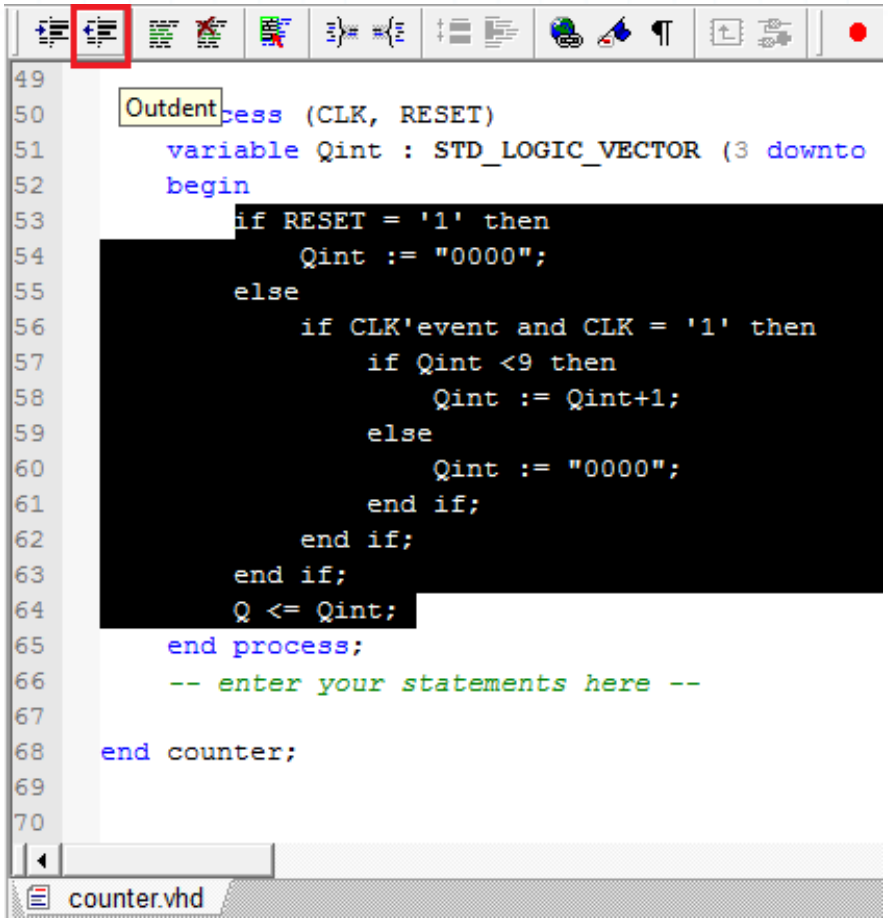
Indenting

- Indenting certain parts within your code is a user-preference for aesthetic purposes.
- There are two ways to indent:
- Select a desired block of code and use the **Indent** icon in the HDE Toolbar.
- Select a desired block of code and press the **Tab** button.



Outdent

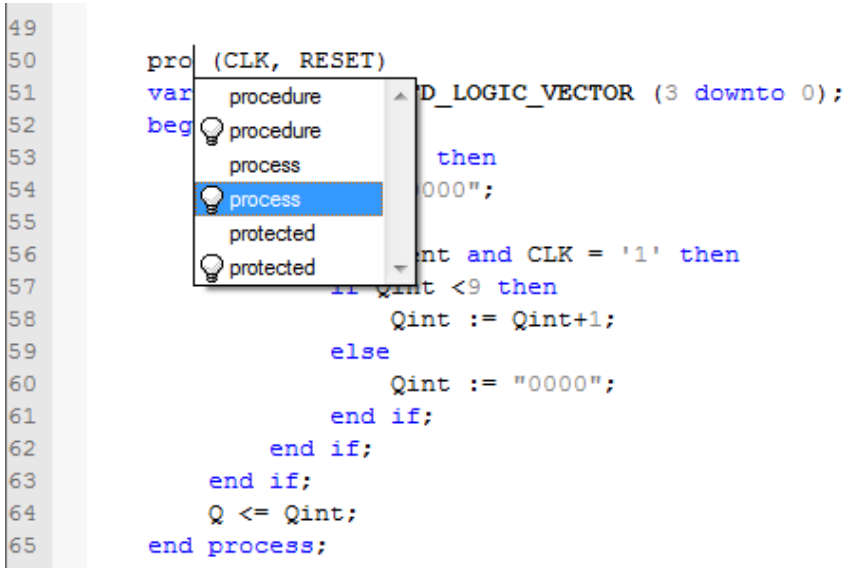
- Similar to Indent, **Outdent** is a user-preference for aesthetic purposes.
- There are two ways to outdent:
- Select a desired block of code and press the **Outdent** icon in the HDE Toolbar.
- Select a desired block of code and press **Shift+Tab**.



Note: You can change the tabulation size in the **HDL Editor** category under **Tools | Preferences**

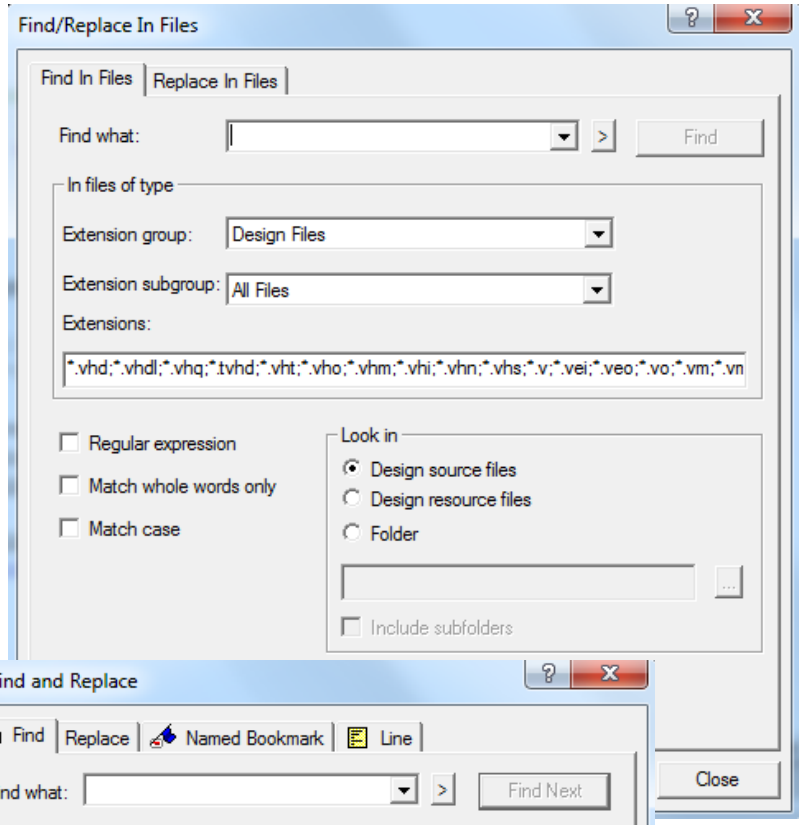
Auto-Complete

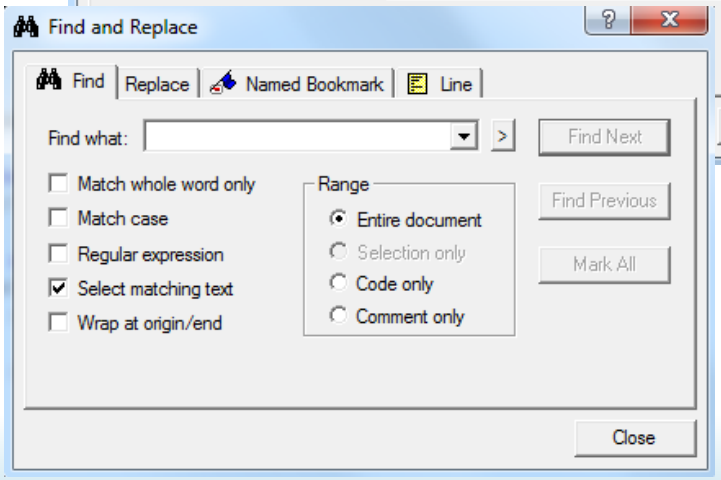
- The HDL Editor automatically completes VHDL and Verilog keywords based on the initial letters that are typed.
- There will be a drop-down box once you start typing. You will be able to scroll through the list of keywords using the arrow keys.
- To select a keyword, press **enter**.



Find

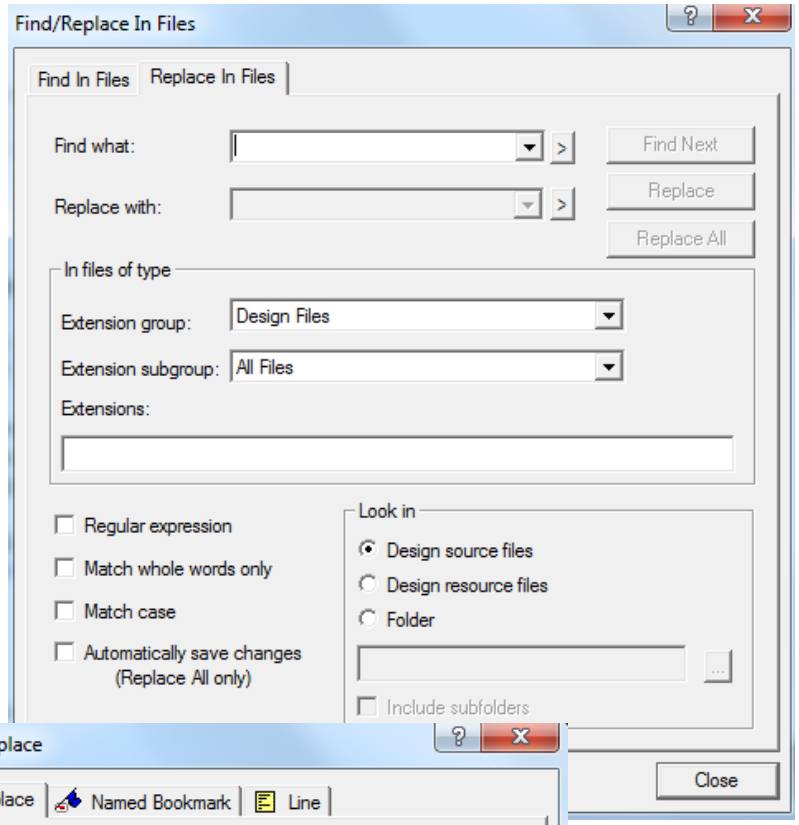
- The **Find** feature allows you to search files for a specific string. It is beneficial for debugging purposes. There are two kinds of **Find** searches:
- **Find**
  - Search for a string in a single file
- **Find in Files…**
  - Search for a string in multiple files at once
- Both can be found under the **Search** menu

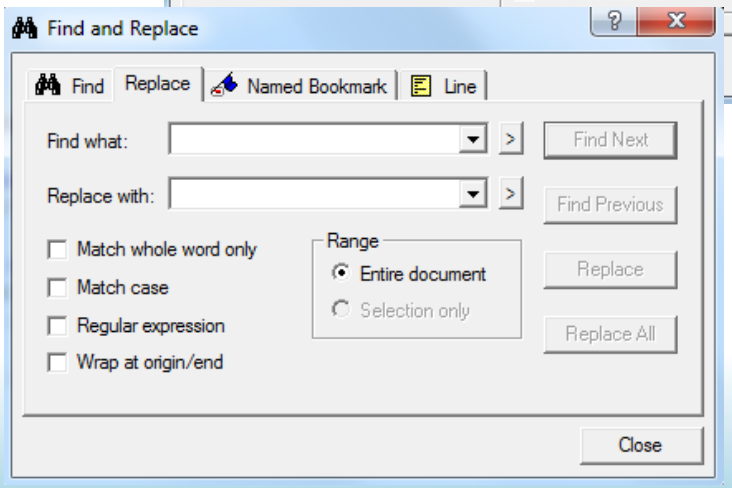




Replace

- The **Replace** feature allows you to search for a specific string and replace the single instance of the specified string or ALL instances of the string with something else.
- Similar to **Find**, **Replace** can also be used for single file searches or multiple file searches.
- The **Replace** feature can be found in the **Search** menu.





Syntax Highlighting

- The HDL Editor support syntax highlighting for the following file

```
entity adder is
    generic ( tpd : time )
    port ( inp : in bit; outp : out bit );
end entity adder;
```

```
vunit OtherEvents(TrafficControl)
{
    default clock is fell(Clk);
    property as_i_p is never {rose(Key and GreenA) ;RedA};
    as_i : assert as_i_p;
}
```

```
 if (++$numopen > $maxopen) {
     splice(@lru, $maxopen / 3);
     $numopen -= @lru;
     for (@lru) { &close($_); delete $isopen{$_}; }
```

```
(SETUP (posedge I) (posedge CLK) (10:20:30))
(HOLD (posedge I) (posedge CLK) (0:0:0))
(SETUP (negedge I) (posedge CLK) (10:20:30))
(HOLD (negedge I) (posedge CLK) (0:0:0))
```

```
 // check 1
 if (vhpi_check_error(&vhpiErrorInfo)){
   switch ( vhpiErrorInfo.severity ){
     case vhpiNote : break;
     case vhpiWarning  : break;
```

- VHDL
- Verilog/SystemVerilog
- PSL/OVA
- Tcl
- Perl
- Macro
- SDF
- EDIF
- C++

```
(interface
    (port IO
        (direction INPUT)
        (property  PINTYPE ( string "IN" ) )
        (property  port_id ( string "3" ) )
    )
```

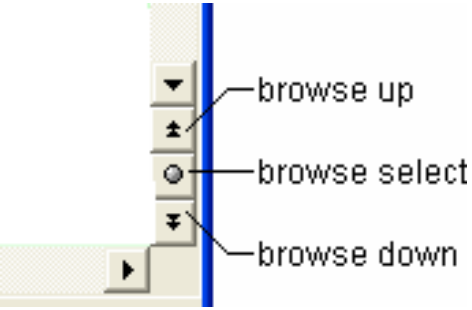
```
proc ExamineAll {} {
global hold bust d_l d_h DisplayTopic;
set hold [examine HOLD];
set bust [examine BUST];
```

```
SetActiveLib -work
comp -include "$dsn\src\counter.vhd"
comp -include "$dsn\src\TestBench\counter_TB.vhd"
asim +access +r TESTBENCH_FOR_counter
wave
```

```
always @(posedge clk) begin
   if (!en)
       out1 <= 1'bz;
end
```

Navigation

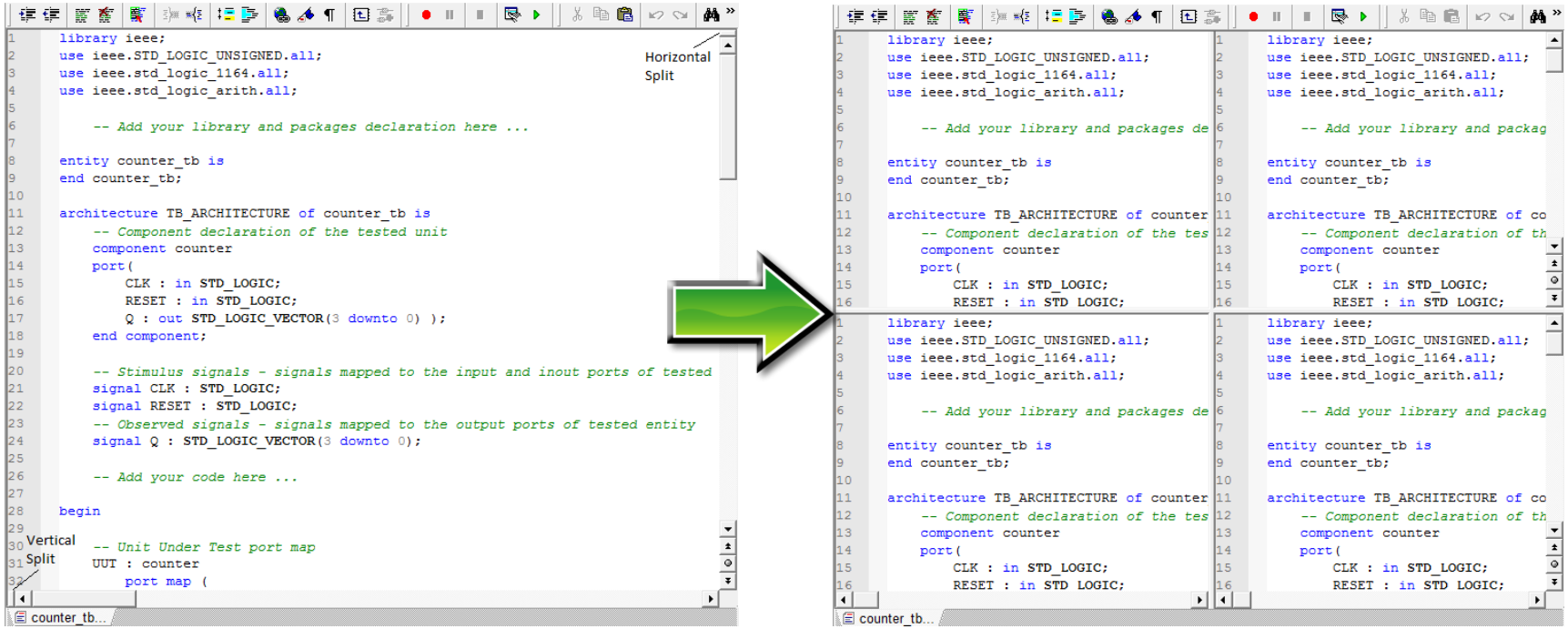
- The HDL Editor provides a number of features designed to facilitate the navigation of source documents in the Active-HDL environment.
- You can browse your code by:
  - Pages
  - Bookmarks
  - Named Bookmarks
  - Links
  - Breakpoints



Browse Options

Navigation (cont.)

- You can create horizontal/vertical split screens to navigate various parts of the same source code.



Bookmarks

- Bookmarks help facilitate navigation through long documents. You can place bookmarks throughout the source code and move from one bookmark to another.
- To insert a bookmark:
- Select the **Toggle Bookmark** or **Ctrl+F2**

Note: All bookmark commands can be found under the Search menu

- To navigate between the bookmarks:
- Select **Next Bookmark** or **F2**
- Select **Previous Bookmark** or **Shift+F2**

- To remove all bookmarks:
- Select **Clear all bookmarks**

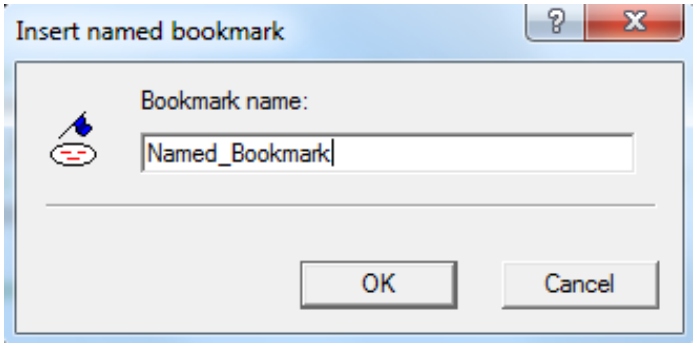
Named Bookmarks

- Named bookmarks are similar to Toggle bookmarks in helping to facilitate navigation through a source code. The difference is that named bookmarks are encoded by special strings inserted directly in the document text. These strings are referred to as *bookmark codes*.
- For example, a bookmark named *jump* will be implemented by the following strings:
  - --<A NAME="Jump">
- The HDL Editor does not display the bookmark codes directly. Instead, the bookmark names are displayed in a distinguished color. Similar to how a comment is displayed.
  - --Named bookmark in HDL code:
  - -- Jump

Named Bookmarks (cont.)

- Any Bookmark codes that occur outside of comments are ignored by the HDL Editor and are displayed as true code.

- To place a named bookmark:
- Select the **Insert Named Bookmark** icon
- Specify the bookmark name




```
20    -- Stimulus signals - signals mapped to the input and inou
21    signal CLK : STD_LOGIC;
22    signal RESET : STD_LOGIC;
23    -- Observed signals - signals mapped to the output ports
24    signal Q : STD_LOGIC_VECTOR(3 downto 0);--Named_Bookmark
25
```

Auto Indent and Smart Indent

- The HDL Editor provides two features designed to facilitate indenting of the edited code:
- Auto Indent
  - When you hit **Enter** to start a new line, the editor automatically inserts tab characters or spaces in the new line to align the insertion point with the first character in the previous line
- Smart Indent
  - When you hit **Enter** to start a new line, the editor automatically inserts tab characters or spaces in the new line to align consecutive HDL constructs
- Both options can be controlled in the **Preferences** dialog.

Auto-Format

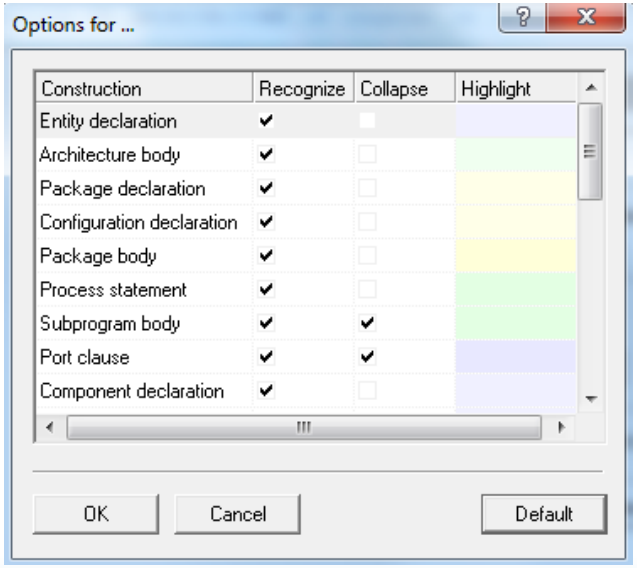
- If you feel like your code is not aligned properly, you can use the **Autoformat Text** feature. This feature analyzes the code and indents consecutive lines of text using the same principles as the **Smart Indent** option.

```
22
23  architecture Counter of Counter is
24      begin
25      -- <<enter your statements here>>
26
27  process (CLK, RESET)
28  variable Qint: STD_LOGIC_VECTOR (3 downto 0);
29  begin
30  if RESET='1' then
31  Qint := "0000";
32  else
33  if CLK'event and CLK='1' then
34  if Qint<9 then
35  Qint:=Qint+1;
36  else
37  Qint:="0000";
38  end if;
39  end if;
40  end if;
41  Q <= Qint;
42  end process;
43  end Counter;
44
```

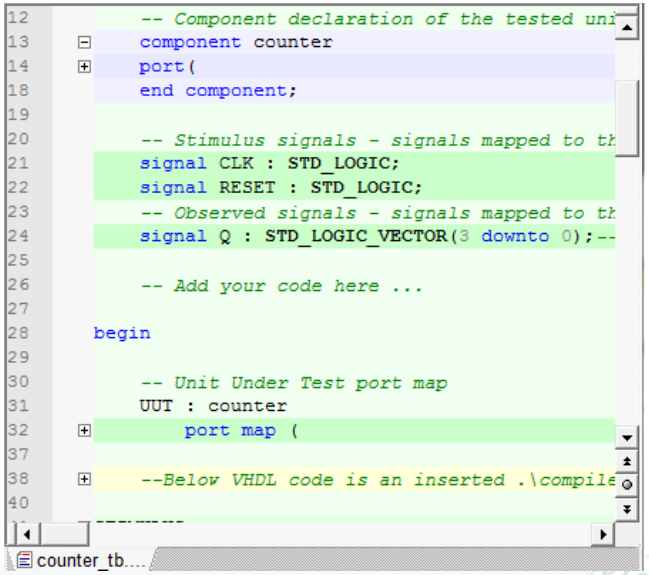
```
23  architecture Counter of Counter is
24      begin
25      -- <<enter your statements here>>
26
27      process (CLK, RESET)
28          variable Qint: STD_LOGIC_VECTOR (3 do
29          begin
30          if RESET='1' then
31              Qint := "0000";
32          else
33              if CLK'event and CLK='1' then
34                  if Qint<9 then
35                      Qint:=Qint+1;
36                  else
37                      Qint:="0000";
38                  end if;
39              end if;
40          end if;
41          Q <= Qint;
42      end process;
43  end Counter;
44
```

Generate Structure

- You can divide your source code into groups using the **Generate Structure** feature. This feature groups parts of code according to the HDL syntax.
- You can customize this command in **Tools|Preferences|Editors|HDL Editor|Languages|<language>|Document structure**



Note: You can customize which HDL constructs should be grouped and the color for each construct



Text Structure

- When you generate a structure for your code, you can use the + - buttons to expand and collapse groups of HDL statements.

- If you do not want your entire source code to be grouped, you can create your own smaller group structures.
  - Select a portion of the code and click the **Create Group** button

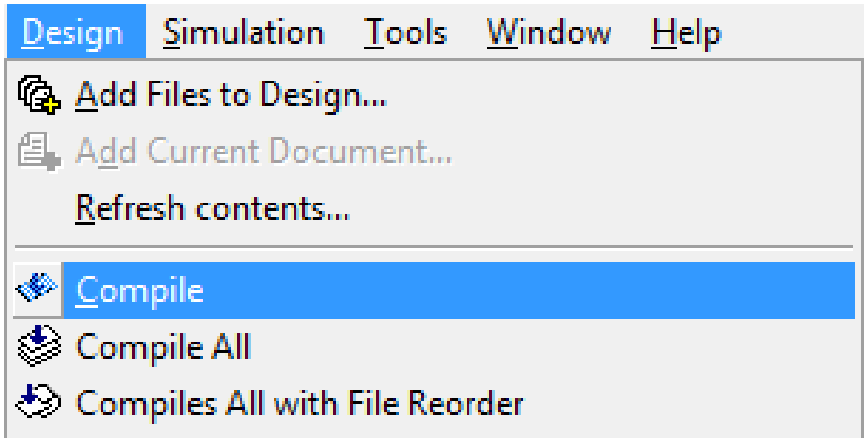
```
        if RESET='1' then
            Qint := "0000";
        else
            if CLK'event and CLK='1' then
                if Qint<9 then
                    Qint:=Qint+1;
                else
                    Qint:="0000";
                end if;
            end if;
        end if;
        Q <= Qint;
    end process;
end Counter;
```

- To revert to the original document layout, click the **Remove Groups** button.

Compiling the Design

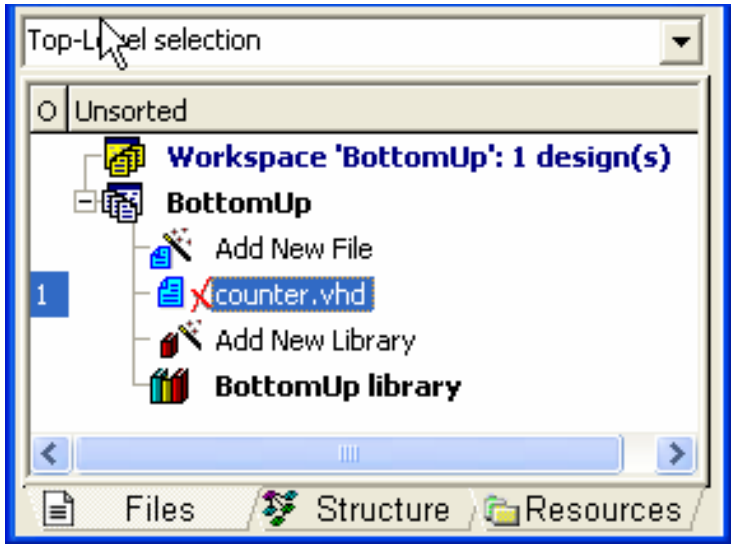
- Active-HDL allows you to compile design source files in three ways:
- Compile
  - Generally for single files
- Compile All
  - For multiple files within a folder or within the entire design
- Compile All with File Reorder
  - Reorders source files to ensure the proper sequence in which design units are compiled

Note: All options can be found under the **Design** menu.



Compilation Status

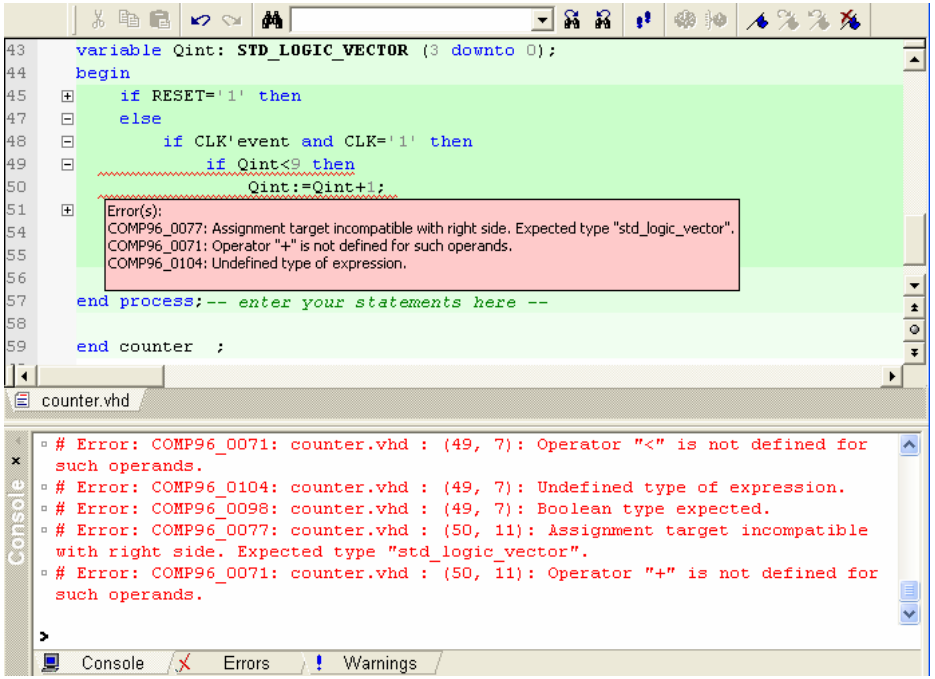
- The status of each compiled filed will be shown in the **Design Browser.** Each source file can have one of the following compilation status markers:
- Errors occurred during the last compilation
- Warnings occurred during the last compilation
- Not compiled or modified after the last compilation
- Successfully compiled



Tracking Errors

- In the **HDL Editor** window, you will be able to track any errors that occurred during compilation (underlined in red).

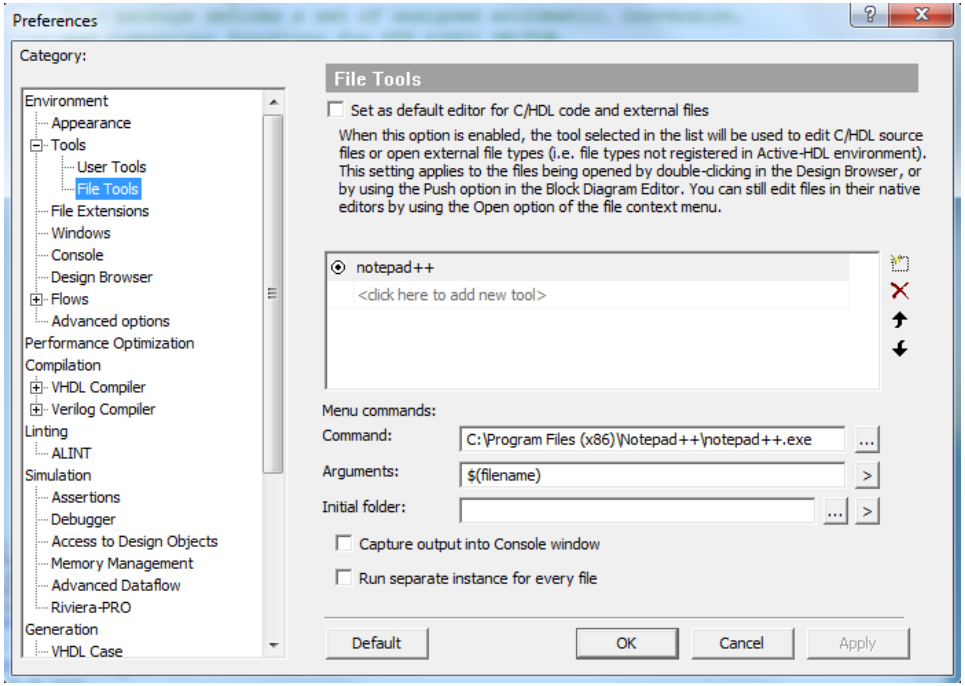
- The **Console** window will display all of the errors with short descriptions. If you double-click on an error, it will take you directly to the error line.



- If you rest your cursor over the underlined error in the HDL Editor, you will also see a short description.

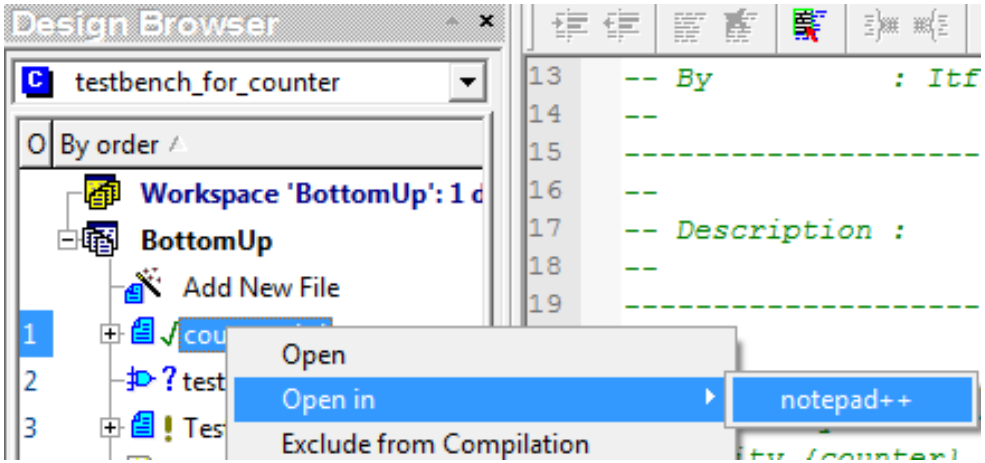
Opening files in External HDL Editors

- Active-HDL allows you to call external editors from within the environment
- To set up:
- Under the **Preferences** dialog, go to **Environment|Tools|File Tools**
- Type the name of the editor where it says **<click here to add new tool>**
- In the **Command** box, point to the location of the .exe file for the external editor.



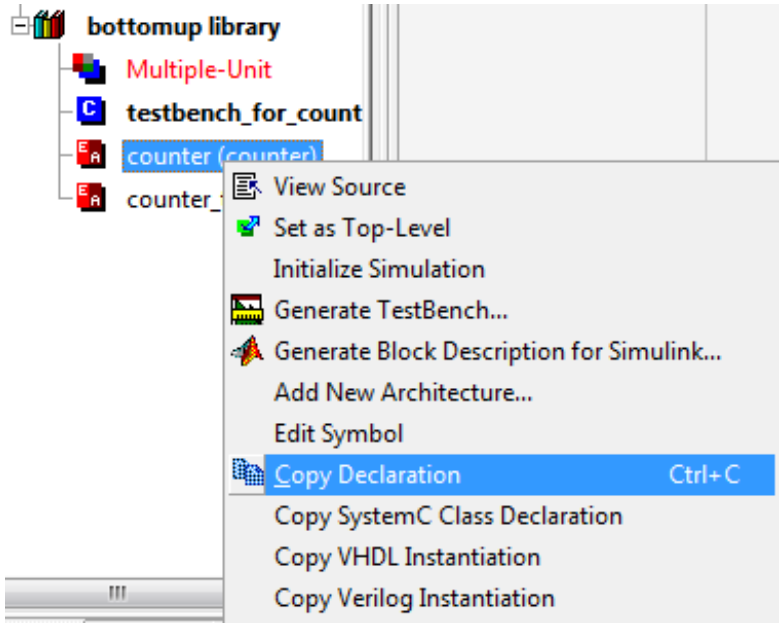
External HDL Editors (cont.)

- To open a source file in the external editor that you created:
- Right-click on the source file in your **Design Browser**
- Select **Open in**
- Choose which editor to open your file



Declaring Components

- Active-HDL provides utilities to speed up a component declaration and instantiation. You can copy a component declaration from the working library or a library in the **Library Manager** window.
- Expand the library contents in the **Design Browser**, right-click the component, and select **Copy Declaration**.
- Go to the HDL Editor window and paste the declaration by right-clicking and selecting **Paste**.



Note: This will only copy the component declaration. You will have to map the ports and generics of the entity by yourself.

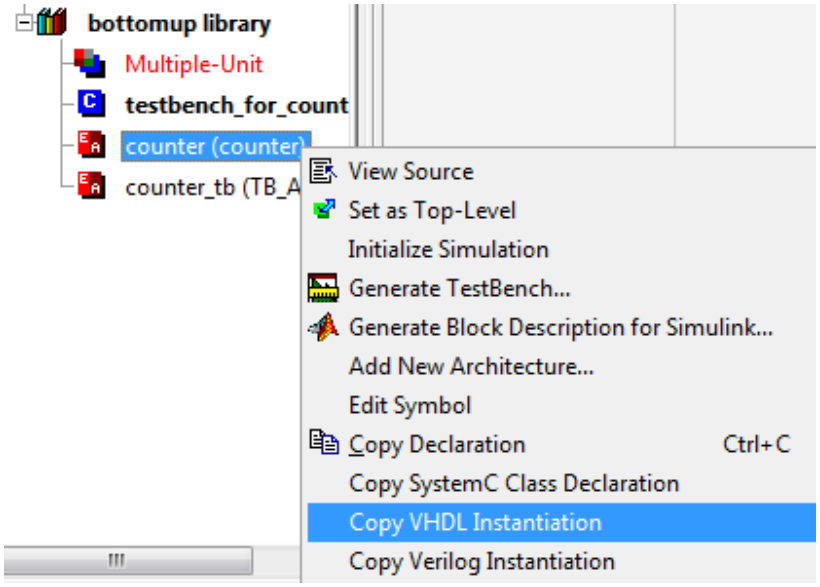
Instantiating Components

- Active-HDL allows you to work with multiple-file projects.
- You can then create required models in separate files and verify them individually instead of placing them in one large design file.
- By creating a top-level entity, you can test the functionality of the entire design. To do this, you have to instantiate the components of the design
- Component instantiation is like plugging a hardware component into a socket in a board.

```
23  architecture MODULATOR of Modulator is
24
25  ---- Signal declarations used on the diagram ----
26
27  signal COS1 : real ;
28  signal SAW1 : real ;
29  signal SIN1 : real ;
30
31  ---- Component declarations -----
32
33  component COSINUSGENERATOR
34    port (
35          CLK : in BIT;
36          CosEnable : in BIT;
37          CosFreq : in INTEGER;
38          COS1 : out REAL
39    );
40  end component ;
41
42  component MULTIPLIER
43    port (
44          IN1 : in REAL;
45          IN2 : in REAL;
46          IN3 : in REAL;
47          clk : in bit;
48          out1 : out real
49    );
50  end component ;
```

Instantiating Components (cont.)

- Active-HDL also provides a feature to speed up a component instantiation. You can derive component instantiation for either VHDL or Verilog.

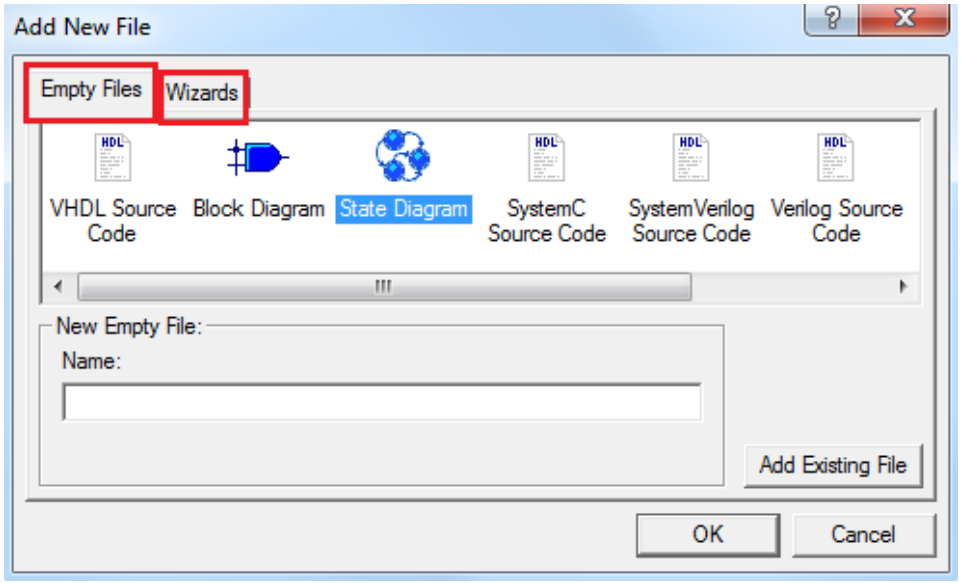


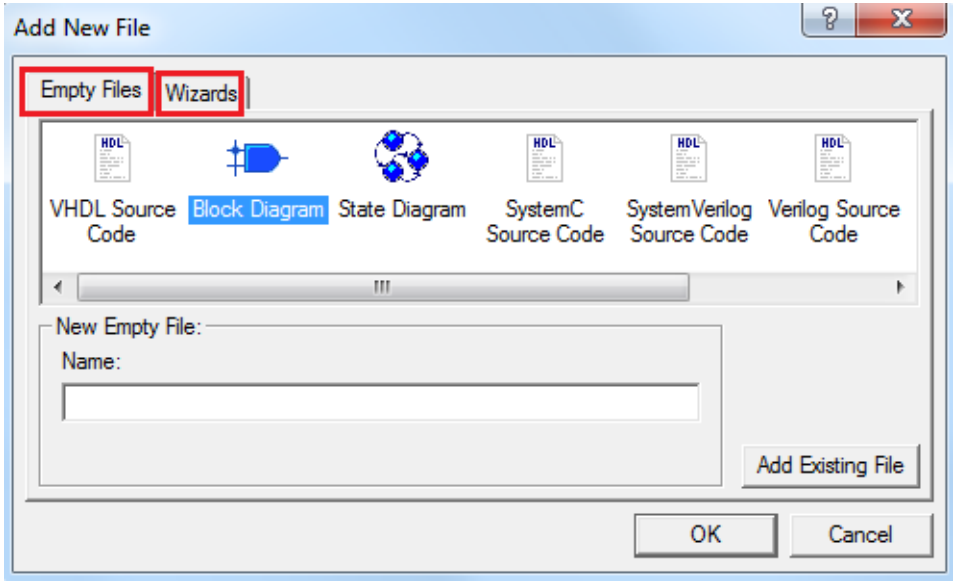
- Expand the library contents in the **Design Browser**, right-click on the component and select **Copy Instantiation** for the appropriate VHDL or Verilog source file.
- Go to the HDL Editor window and paste the instantiation by right-clicking and selecting **Paste**.

Note: The instance label and actuals mapped to the ports of the instance can be customized in the **Preferences** window.

Creating State Machines

- To create a **State Diagram**:
- Double-click **Add New File** in your **Design Browser**.
- You can choose to create a **State Diagram** without any assistance under the **Empty Files** tab.
- Or you can choose to create a **State Diagram** with a step-by-step guide under the **Wizards** tab.



Creating Block Diagrams

- To create a **Block Diagram**:
- Double-click **Add New File** in your **Design Browser**.
- You can choose to create a **Block Diagram** without any assistance under the **Empty Files** tab.
- Or you can choose to create a **Block Diagram** with a step-by-step guide under the **Wizards** tab.