HDS Language Support Guide

Software Version 2010.3

June, 2011

Contractor/manufacturer is:
Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: www.mentor.com
SupportNet: supportnet.mentor.com/
Send Feedback on Documentation: supportnet.mentor.com/user/feedback_form.cfm

Table of Contents

Chapter 1
SystemVerilog Support

Introduction

SystemVerilog combines the Verification capabilities of HVL (Hardware Verification Language) with ease of Verilog to provide a single platform for both design and verification.

HDS currently provides text-level support for SystemVerilog 3.1a/1800 standard. SystemVerilog files/designs can be created in HDS, organized into libraries / projects and managed accordingly. Not only can you create new designs but also you can import any of your old SystemVerilog code created outside HDS. Doing so you can make use of all design management capabilities provided through the tool.

HDS Design Manager helps you browse through your code and view it from different perspectives. It also allows you to obtain reports on specific aspects of your design as assertions, coverage points, etc.,

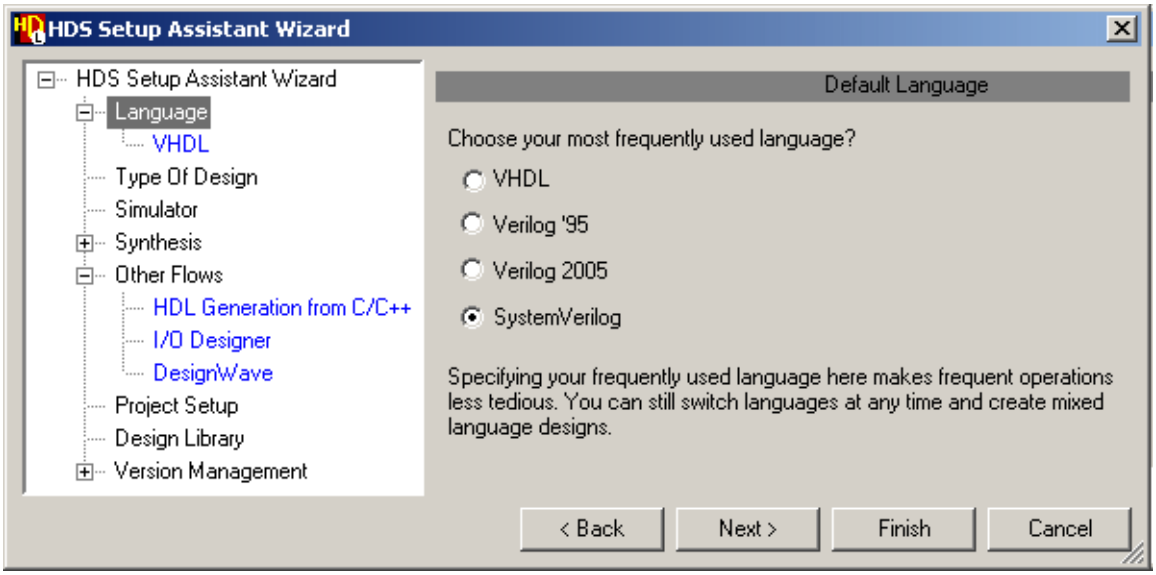
In this chapter we provide a quick tour through the different features and tools available in HDS that you can use in developing your SystemVerilog designs.

Setting SystemVerilog as your Default Design Language

HDS allows you to set SystemVerilog as the default language for all newly created designs. This can be helpful if SystemVerilog is your most frequently used design language.

Procedure

1. Choose **Help>HDS Setup Assistant**.
2. Select the Language Node and set the SystemVerilog option in the left pane of the window. Click **Finish** to save your new setting.



Related Topics

- HDS Setup Assistant Wizard
- Main Settings Dialog

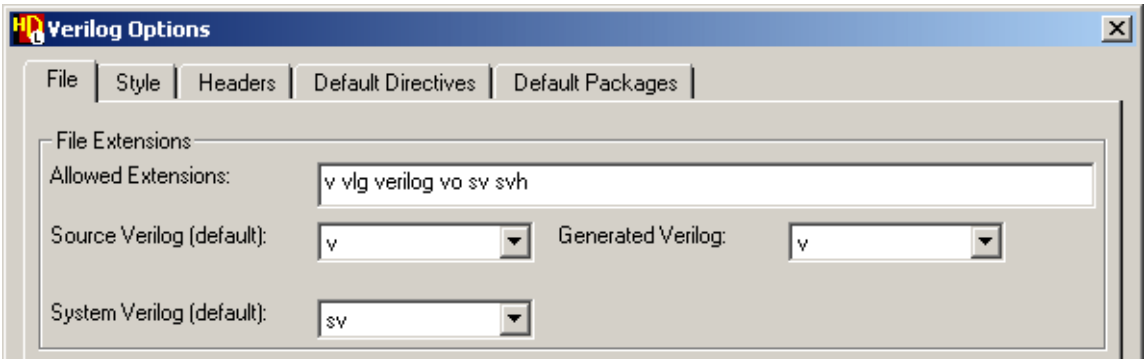
Setting SystemVerilog Language Preferences

New SystemVerilog designs are created according to a set of predefined options. The options are related to file naming, file headers, compiler directives and default packages.

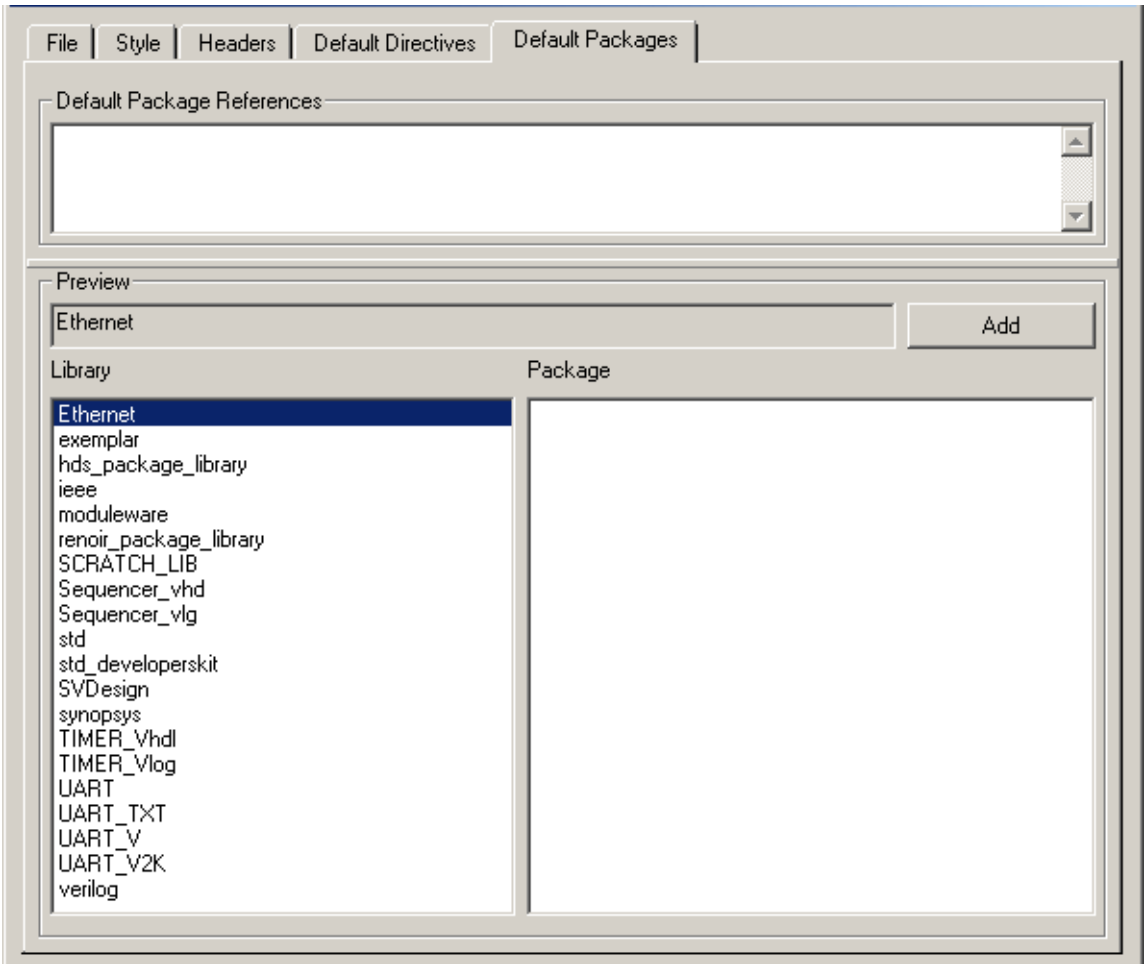
Procedure

1. In the **Design Explorer** window select **Options>Verilog** to display the **Verilog Options** dialog.

2. On the **Files** tab specify your default Verilog file extension by choosing from the System Verilog(default) dropdown list.



3. On the **Headers** tab choose from a pulldown list of view headers. The selected header is is used in the created view. Note that graphical views are not supported.
4. You can set default Verilog compiler directives in the **Default Directives** tab.
5. You can set default packages in the **Default Packages** tab.



6. Click **OK**. The above defined preferences will be added to new views.

Related Topics

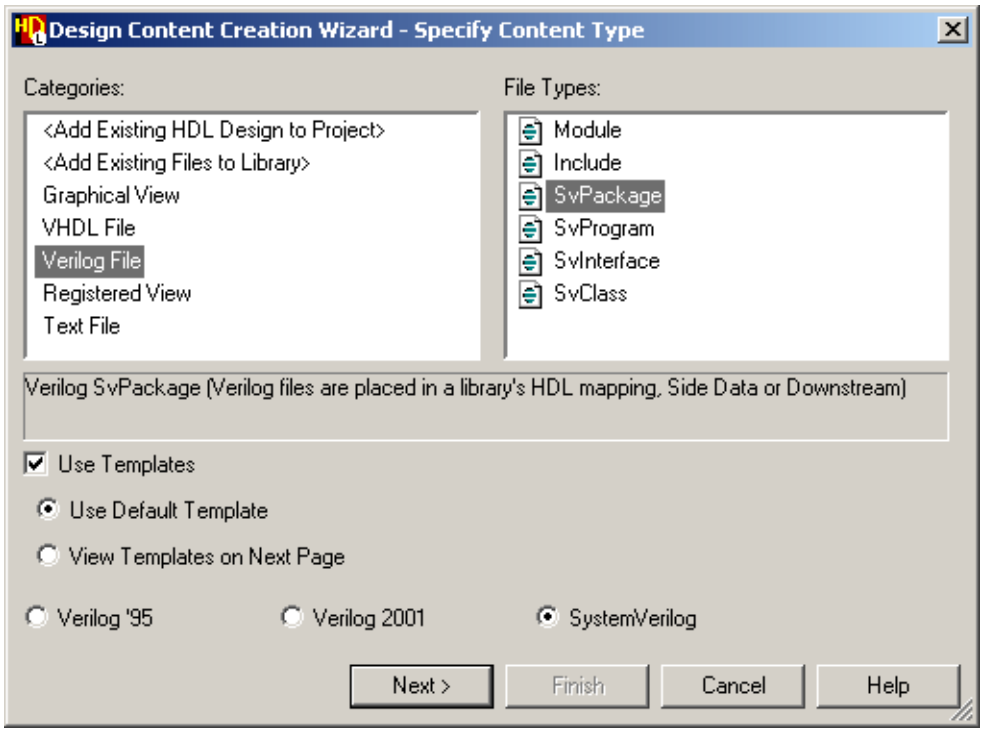
- Preferences for HDLViews

Creating SystemVerilog Files

HDS allows you to create text SystemVerilog views. Preferences set in the **Verilog Options** dialog are applied to the created views

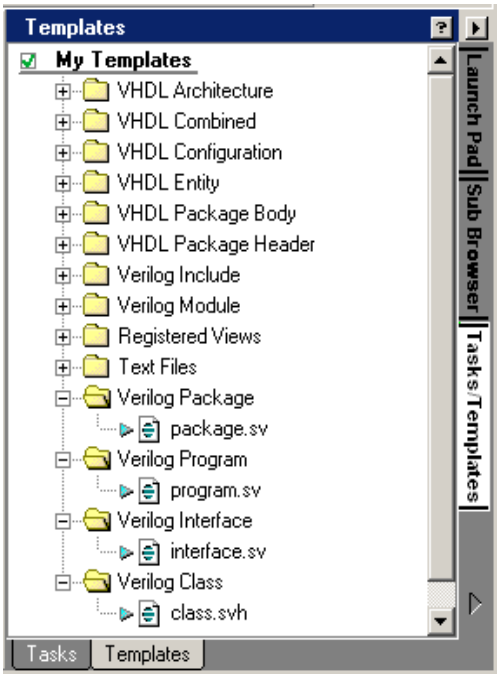
Procedure

1. Invoke the **Design Content Creation Wizard**.
2. Choose Verilog File from the Categories pane. In the File Types pane specify your file type as Module, Include, SV Package, SV Program, SV interface or SVClass.



3. Set the language dialect as SystemVerilog.

If you choose to use a template define whether you want to specify your own or work with the default one. Templates are available for SV packages, programs, interfaces and classes.



4. Click **Next** to specify the new Verilog file location, name and format.
5. Click **Finish**. The DesignPad editor is launched displaying your new SystemVerilog file.

   You can check your new design object in the Design Units pane of the Design Explorer.

Tip: To display a created class in the Design Unit Browser pane add your code after the module code or create a new class from the Design Creation wizard. Nested Classes only appear in the DesignPad code browser.

Related Topics

- Using the Design Content Creation Wizard

Adding SystemVerilog Designs

Existing SystemVerilog designs can be added to HDS projects through the Add Existing Design wizard. According to your preference you can move a copy of the design to the HDS projects' directory or point to it in it's current location.

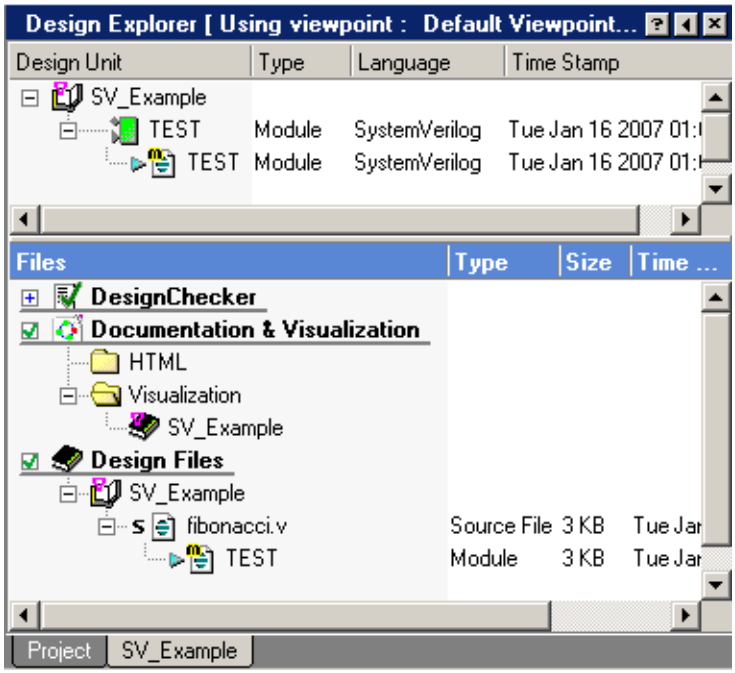
Procedure

1. Start the **Add Existing Design** Wizard by choosing **File>Add>Existing Design**:
   a. Specify the method to add design content by choosing **Copy Specified Files or Point to Specified files** from the Add Method group box.
   b. Specify whether you would like to use a filelist to obtain information on design content by setting the Use Filelist option. Specify the files to add.
2. Specify the method to determine the Verilog dialect used as auto or Specified.

Table 1-1. Verilog Dialect Setting Options

Option	Description
Auto	The Verilog dialect used is automatically detected. A single design may include different verilog dialects
Specified	The verilog dialect is marked as SystemVerilog for all design files even if they do not contain any SystemVerilog constructs.

3. Complete the **Add Existing Design** wizard steps. The Design Explorer window is displayed showing the added design.



Related Topics

- Adding Existing Design Content to a Project
- Adding Design Files to a Library

- Removing Design Content
- Add Existing Design Wizard

Editing SystemVerilog text Views

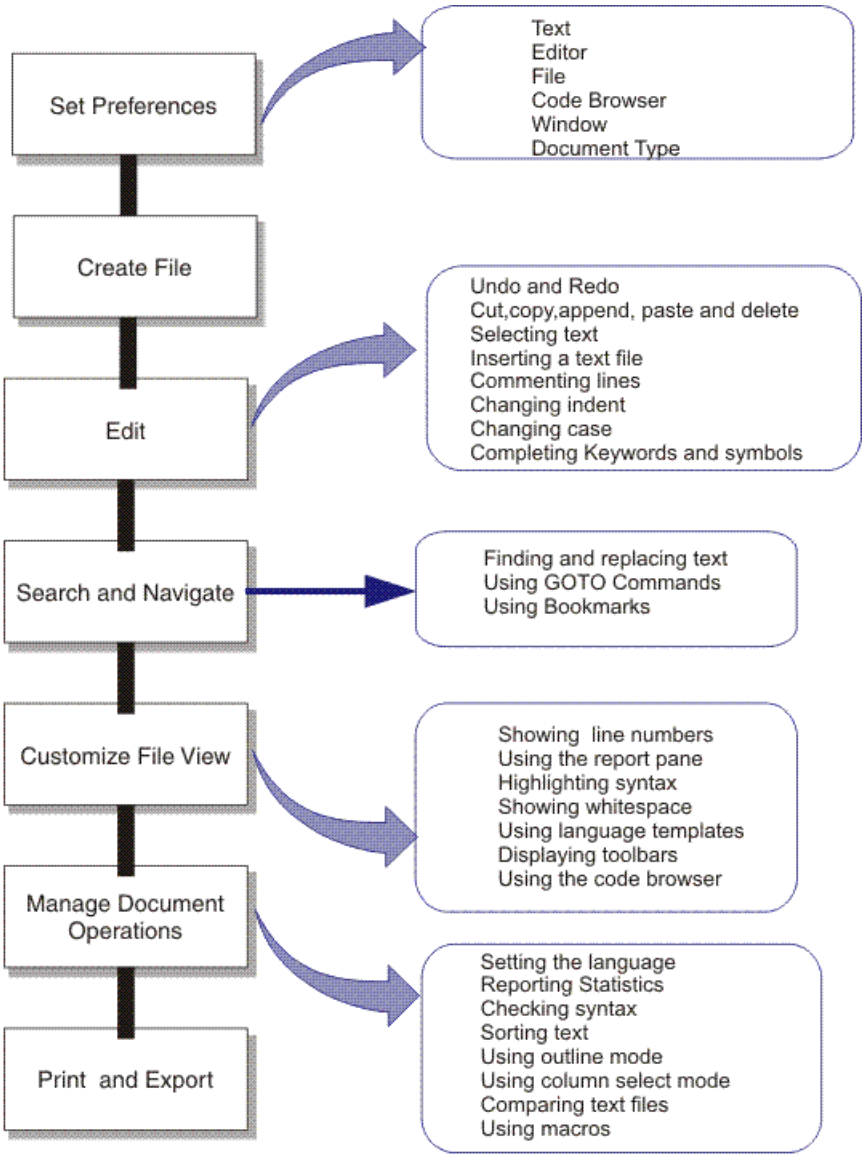
All normal text editing procedures provided through DesignPad are supported for SystemVerilog views. HDS allows you to set language specific preferences as syntax highlighting and code browser content.

When editing SystemVerilog documents SystemVerilog Keywords are automatically detected and completed by choosing **Complete Keyword** from the **Edit** or popup menu.

Refer to *About DesignPad*.

The figure below illustrates how HDS allows you to manage your design files in DesignPad.

Figure 1-1. Managing Design Files in DesignPad



Setting SystemVerilog Syntax Highlighting Preferences

1. In DesignPad, choose **Options>Preferences**. The Preferences dialog box is displayed.
2. In the Categories pane browse to Document Type>SystemVerilog>Syntax Highlighting to display the Syntax Highlighting page.
3. Set the Enable Syntax Highlighting option.
4. To change the default syntax highlighting options for an object, select the required object from the list in the Syntax group Box and define the new object background and foreground colors.

Related Topics

- SystemVerilog Syntax Highlighting

Setting Code Browser Content for SystemVerilog

The code browser represents the code structure in the active file as a hierarchical tree of recognized code blocks i.e objects. You can set the SystemVerilog objects you wish to display in the code browser.

Icon	Code Block	Icon	Code Block
	Always		External Instance
	Always_Comb		Interface
	Always_FF		Interface Port
	Always_Latch		Local Parameter
	AssertProperty		Modport
	AssumeProperty		Module
	Class		Package
	ClassObject		Parameter
	Clocking		Parameter Map
	Const		Parallel Block
	CoverageBin		Package Import
	CoverageCross		Input Port
	CoveragePoint		Inout Port
	Covergroup		Port Map
	CovergroupObject		Output Port
	CoverProperty		Program
	Cross		Property
	ExpectProperty		Sequential block
	ExportImport		Sequence
	Final		Specify
	Function		Specific Parameter
	Gate Instance		Struct
	Generate Statement		Task
	Generate Variable		Type

Icon	Code Block	Icon	Code Block
	Include		Wire
	Initial		Union
	Instance		

Procedure

1. In DesignPad, choose **Options>Preferences**. The **Preferences** dialog box is displayed.
2. In the Categories pane browse to **Document Type>SystemVerilog>Code Browser**.

   The Code Browser page is displayed showing a list of all the available SystemVerilog objects.
3. Select/deselect the objects you wish to show/hide in the code browser window.

Exploring SystemVerilog Designs

SystemVerilog designs can be explored and managed through HDS Design Manager. The Design Manager provides 3 main views of the HDL design data. Each view is displayed in a distinct design browser pane. You can use the cross-highlight feature to track down an object in the different design views.The table below list the three main Design Manager views:

Table 1-2. HDS Design Managers Views

Design Manager View	Description
Design Unit	This is a “virtual” representation of the main declarative objects within the design.
Design Files	This shows the actual file structure of the files and folders which are part of the project. HDL source files can be expanded to see the top-level objects within them.
Design Hierarchy	This shows the static instance hierarchy from a given level. Modules, classes and Program Blocks can be dragged into this browser in order to explore the design hierarchy underneath that level.

Tip: The Design Unit browser can be toggled to an Object List browser where you can group, sort and filter all objects (including instances)

You can control which objects are visible, which columns of additional data are shown (including user defined property information) and save these settings in alternative “viewpoints”.

Related Topics

- Using Design Explorers
- Using the Side Data Browser

Identifying SystemVerilog Design Objects

The following SystemVerilog Objects are displayed in the Design Explorer browsers.

Program Blocks

Program blocks are intended to contain test bench code i.e they relate to verification only.They represent leaf-levels of hierarchy.

Interfaces & Modports

Interfaces are a special kind of port type but can be treated in a similar way to Modules.

Interfaces can

- Describe a re-usable group of signals. Modports allow this overall group of signals to be divided into a number of subgroups and also have specific directionality for example – master and slave.
- Model “protocol” behavior, including protocol checking when associated with tasks and functions.
- Be used as “adaptors” or “converters” thus allowing the same component to be connected up at differing levels of abstraction.

Classes

Classes are user-defined data types that bring in the essence of object oriented programming to the SystemVerilog language.

Tip: Systemverilog only allows one level of inheritance.

Packages

A package is a new SystemVerilog design unit similar to the VHDL package.

It is used for

- Sharing declarations among modules, interfaces, programs and other packages.
- Defining a single global set of items that can be used by any design unit that imports that package.

SystemVerilog Assertions

An assertion is a statement that validates assumptions or checks conditions in a program

SystemVerilog has an integrated set of constructs that help you to build assertions and closely couple them with the rest of your design or verification code.

SystemVerilog assertions can

- Provide control-oriented coverage. Coverage allows you to tell how well a design has been tested.
- Specify and validate design behavior.

Cover groups

Cover groups, derives from hardware verification languages. Cover groups can contain multiple individual coverage points, allow "binning" of multiple values and support cross-coverage to track combinations of values.

In the following section we will quickly view how the new SystemVerilog objects are displayed in each Design Explorer browser.

Design Unit Browser

The key new information/objects supported in the Design Unit browser are:

Table 1-3. System Verilog Design Unit Browser Objects

Icon	Description
	Program Block
	Interface
	Class
	Package
	Module
	Include

Files' Browser

In the Files' Browser HDL source files can be expanded to see the top-level objects within them.The key information/objects supported in the Files' browser for Systemverilog designs are:

Table 1-4. System Verilog Files' Browser Objects

Icon	Description
	Program Block
	Interface
	Class
	Package
	Module

Design Hierarchy Browser

The Design Hierarchy browser shows static design hierarchy. Unresolved objects are highlighted.

Related Topics

- Finding Design Objects
- Finding Unbound Components
- Finding Unparsed Files

Reporting Assertions and Covergroups

SystemVerilog provides two basic mechanisms for specifying functional coverage. The first, cover groups and the second cover properties. The cover property construct is part of the SystemVerilog Assertions (SVA) subset, sharing temporal sequences and other building blocks with assertion specifications.

Specific reports to summarize and locate assertions, property-based coverage and cover groups can be generated through HDS.

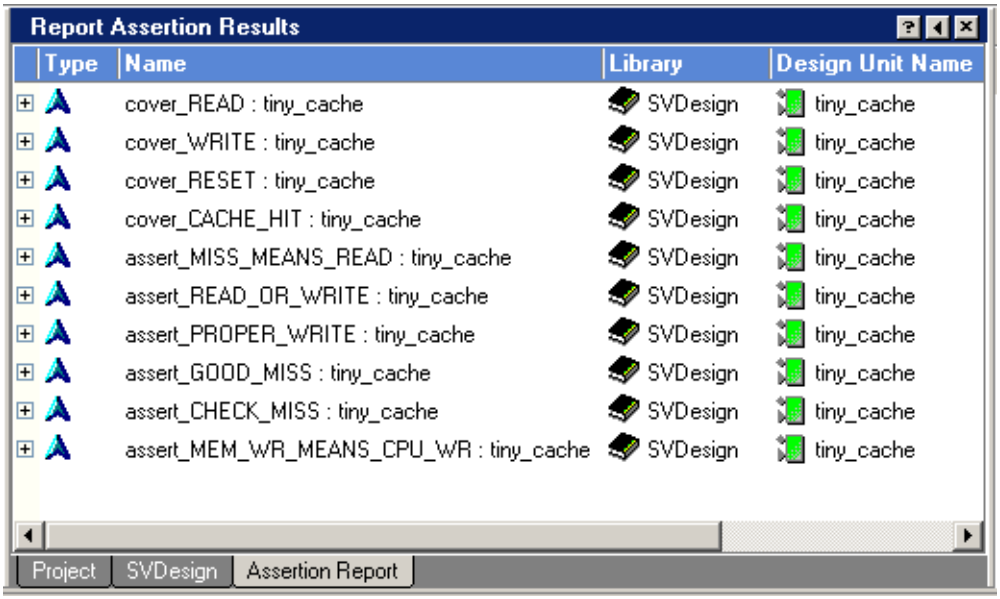
Procedure

In the Design Manager window

1. Select a module and from the popup menu choose **Reports>Assertions (SV)** or **Reports>Covergroups(SV)**

You can report assertions/coverage within a single object, its hierarchy or within the entire library.

A new tab is displayed showing a report of the design assertions/covergroups.



Related Topics

- Finding Design Objects
- Finding Unbound Components
- Finding Unparsed Files

Reporting SystemVerilog Bound Objects

Although test bench constructs can be embedded within the functional description, they are often written in separate verification files which are then associated with the functional description (typically using the keyword BIND)

The Bind statement allows the designer (or verification engineer) to associate reusable verification components (e.g. properties and assertions) with a design without changing the source code. You can bind a module, interface or program instance to a module or module instance.

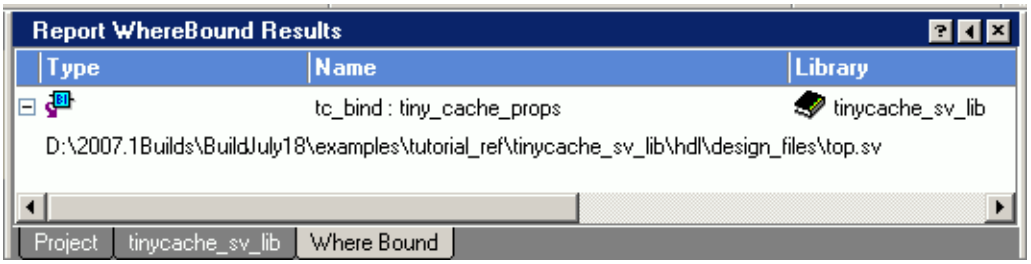
You can create a report showing all the files to which a selected module is bound.

Procedure

In the Design Manager window

1. Select a module and from the popup menu choose **Reports>Where Bound**

   A new tab is displayed showing a report of all the files to which a selected module is bound.



2. Click on any of the file entries. DesignPad opens and highlights the bind statement.

Compiling and Simulating SystemVerilog Designs

SystemVerilog designs can be compiled and simulated using ModelSim/QuestaSim.

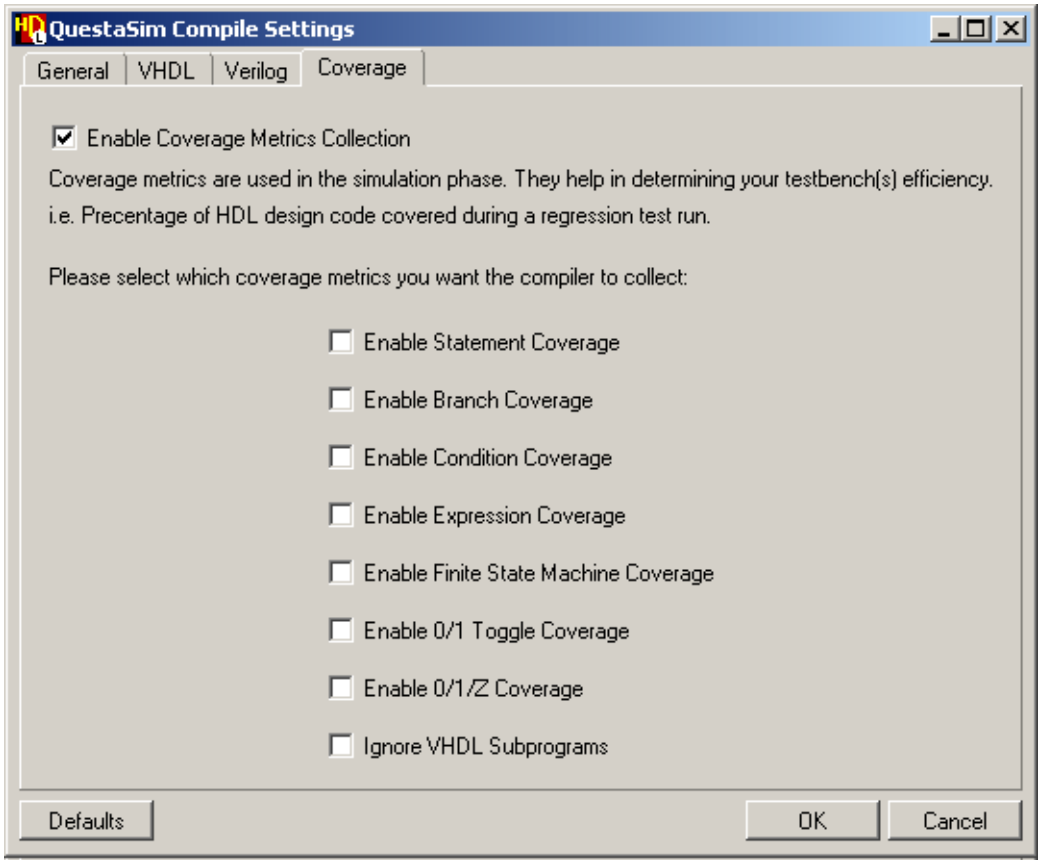
Prerequisites

ModelSim/QuestaSim should be installed.

Procedure

1. Choose **Help>HDS Setup Assistant** menu to display the HDS Setup Assistant Wizard.
2. Open the **Simulator** page which automatically detects the presence of any installed simulators and displays the path of the executables; select the Model*Sim* simulator or the QuestaSim simulator.
3. Click **Finish**.
4. Select the root design unit in the Design Unit Browser.

5. Choose **Settings** from the popup menu of the QuestaSim Compile task to display the **Compile Settings** dialog. Check the Enable code coverage option on the coverage tab.



6. Choose **Run through Components** from the popup menu of the Simulator button in the shortcut bar.

   The progress of the compilation is shown in the HDL Log Window.

Tip: Compilation warning messages may appear in the log window.

7. On completion of the compilation of the design, the Start Simulator dialog box is displayed.
8. Accept the default resolution and check that the **Interactive (GUI)** and **Enable Communication with HDS** options are selected.
9. Check the Enable code coverage option.
10. In the Additional simulator arguments field enter

    ```
    -novopt-assertdebug
    ```

    To suppress the vopt optimization mode and enforce the debug mode.

11. Use the **OK** button to confirm the Start Model*Sim*/Questa*Sim* dialog box and load the design

Chapter 2
Verilog 2005 Support

Introduction

HDS provides both text-level and graphical support for the Verilog 2005 language.In this chapter we will provide an overview of the features in HDS designed to support the V2k language.

Importing existing Verilog 2005 designs in addition to creating new designs is now possible.You can even toggle the language of existing HDS Verilog 95 designs.

Verilog 2005 text designs can be instantiated inside Block Diagram and IBD views as well as visualized or converted to graphics. Generate constructs are recovered as Embedded Blocks. HDS does not currently provide support for Generate Frames in Verilog 2005 designs.

Signed signals and ports, uwire net types, declarations with continuos assignments in addition to initialized variable declarations are now supported.

Verilog 2005 style code can be generated from HDS Graphical Editors. HDL created or generated code can be compiled and simulated.

Setting Verilog 2005 as the Default HDS Language

HDS allows you to set Verilog 2005 as the default language for all newly created designs. This can be helpful if Verilog 2005 is your most frequently used design language.

Procedure

1. Choose **Options>Main** from the Design Explorer window. The **Main Settings** dialog appears
2. On the **General** page, in the Default Language for New Views group box choose the Verilog 2005 option.

Tip: You can change the set default language for a specific view from the Design Content Creation wizard.

3. Choose **Help>HDS Setup Assistant**.
4. Select the Language Node and set the SystemVerilog option in the left pane of the window. Click **Finish** to save your new setting

You can change the set default language for a specific view from the Design Content Creation wizard.

Related Topics

- HDS Setup Assistant Wizard
- Main Settings Dialog

Migrating your Verilog 95 Designs to Verilog 2005

Converting your old Verilog 95 designs to Verilog 2005 and making use of all the supported features may sound like a very tempting idea. The only obstacle you may be thinking of is the time you have to invest in such a task. The good news is HDS allows you to toggle the language of your Verilog95 designs(Graphical and text) in a very simple and easy way without having to redo your work.

Changing the language of your design unit automatically changes the language of the associated symbol as well as all the views representing this design. Refer to "Procedure" on page 39.

The HDL code generated from the converted designs will follow the set Verilog 2005 generation properties.

Prerequisites

Existing Verilog 95 design.

Caution

VHDL or SystemVerilog designs can not be toggled to Verilog 2005 ones

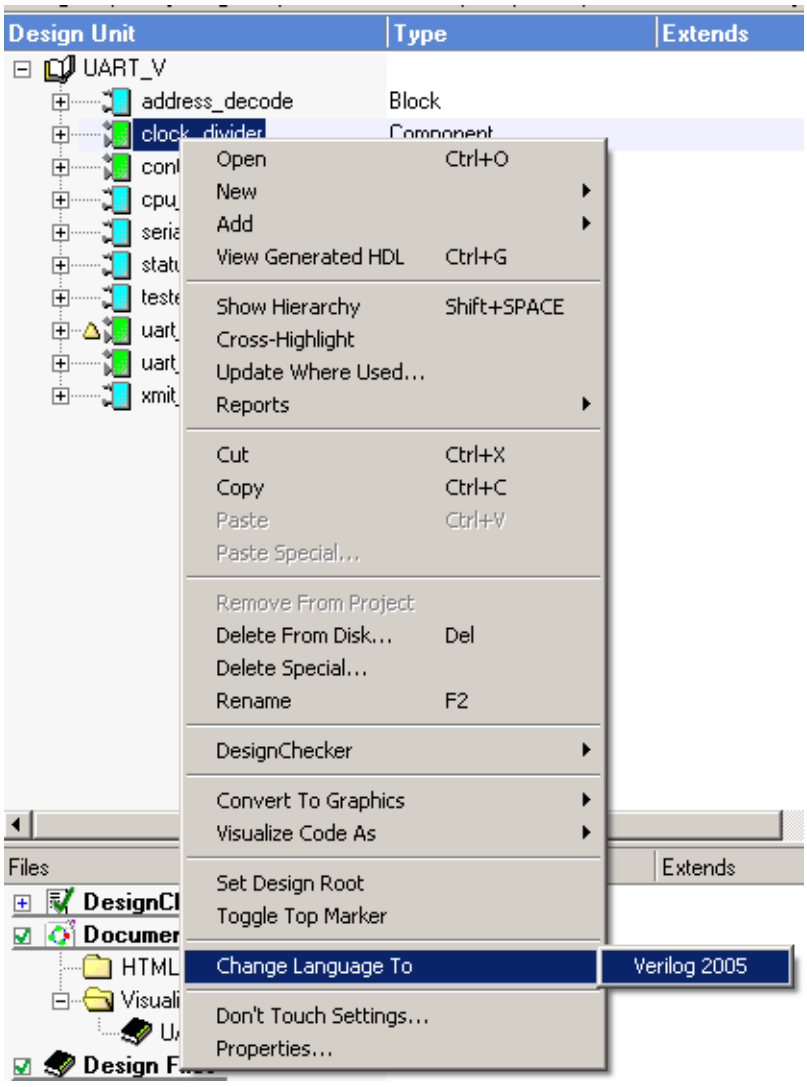
Procedure

1. Select a Design Unit from the Design Units pane in the Design explorer window and choose **Change Language to>Verilog 2005** from the popup menu.

   Notice that the Language field now displays Verilog 2005 instead of Verilog 95 for the selected design unit.

Caution

Verilog 2005 graphical views can not be toggled back to Verilog 95 ones.



Example

This is an example to show how you can toggle the language of your existing Verilog 95 designs and make use of the new Verilog 2005 supported features. It also points out the rules followed when changing the language to Verilog 2005.

Prerequisites

Create a new library with the name backup and copy the contents of the Uart_V library into it.

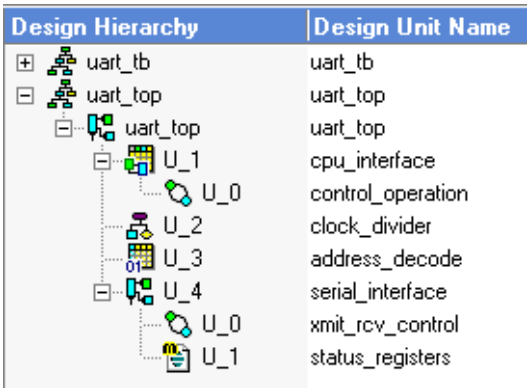
Steps

1. Open the Uart_V library from the **Project Manager** Window.

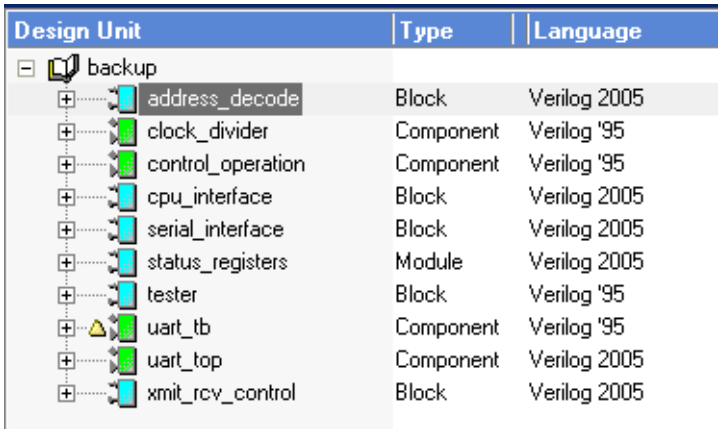
2. In the Design Units browser select the uart_top design unit and choose **Change Language To>Verilog 2005** from the popup menu.

   Notice that the Language field now displays Verilog 2005 instead of Verilog 95 for the selected design unit.

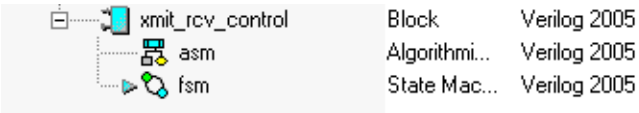
3. Drag the uart_top design unit to the hierarchy browser. Examine the displayed hierarchy.



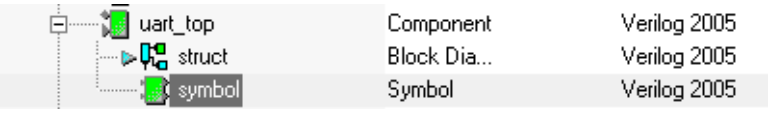
4. Notice how the language of all Blocks and Embedded Blocks which exist under the uart_top design unit has been changed to Verilog 2005. Toggling the language of a Block Diagram or IBD view enforces the toggling of all underlying Blocks and Embedded Blocks.



5. Expand the xmit_ rcv_control design unit. The language of all views for this design unit have been automatically changed to Verilog 2005.



6. Expand the uart_top design unit to view its symbol. The symbol language has been automatically changed to Verilog 2005.



7. Select the uart_top symbol and choose **Open** from the popup menu. A graphical representation of the symbol is displayed. In the Structure Navigator bar double click **Interface** to display a tabular representation of your symbol

   Notice the new Signed and Value columns.

	A	B	C	D	E	F	G	H
A	Group	Name	Mode	Type	Signed	Bounds	Value	Comment
1		addr	input	wire		[2:0]		
2		clk	input	wire				10 MHz clock
3		cs	input	wire				chip select
4		nrw	input	wire				read(0), write(1)
5		rst	input	wire				reset(0)
6		sin	input	wire				serial input
7		int	output	wire				interrupt (1)
8		sout	output	wire				serial output
9		datin	input	wire		[7:0]		data in bus from
10		datout	output	wire		[7:0]		t data out bus to
11								

8. Expand the uart_top design unit to view its symbol. The symbol language has been automatically changed to Verilog2005.

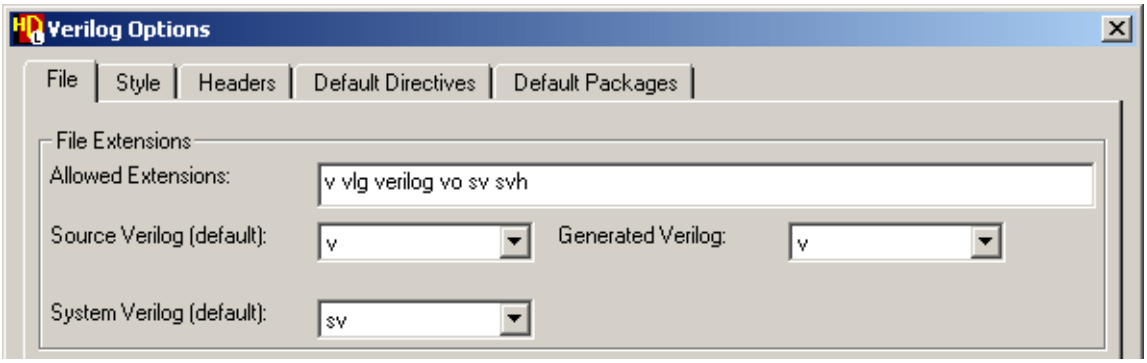
Setting Verilog 2005 Language Preferences

New Verilog 2005 designs are created according to a set of predefined options. The options are related to file naming, file headers and compiler directives and are set through the Verilog Options dialog.

Procedure

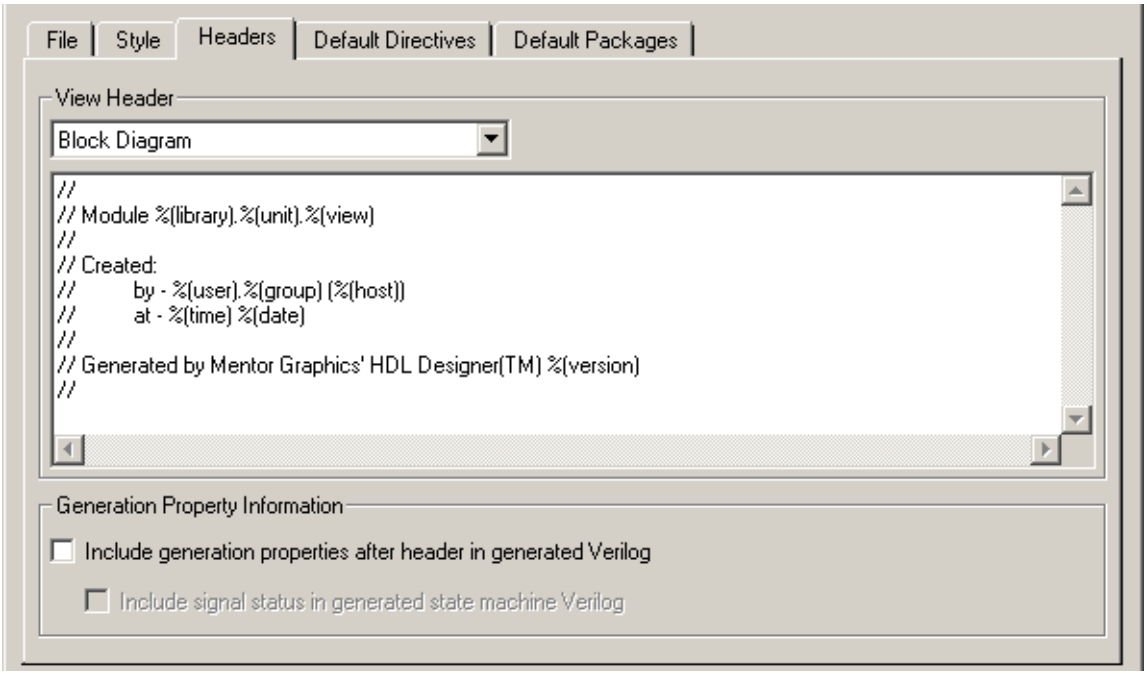
1. In the **Design Explorer** window select **Options>Verilog** to display the **Verilog Options** dialog.

2. On the **Files** tab specify your default Verilog file extension by choosing from the Source Verilog(default) dropdown list.



3. On the **Headers** tab choose from a pulldown list of view headers. The selected header is is used in the created view.

   You can also choose to include the generation properties as comment text after the header in the generated HDL and if this option is set, you can also include the signal status in generated state machine HDL



4. You can set default Verilog compiler directives in the **Default Directives** tab.
5. Click **OK**. The above stated preferences will be added to any new view.

Working with Verilog 2005 Interfaces

An HDS interface defines the output and input ports for a design. The interface is presented in either a tabular form and is referred to as an Interface or a graphical form and is referred to as a Symbol.

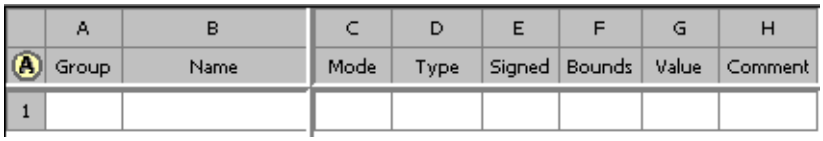
Creating a Verilog 2005 Interface

Component interfaces are created through the **Design Content Creation** wizard and edited through the tabular IO and symbol editors.

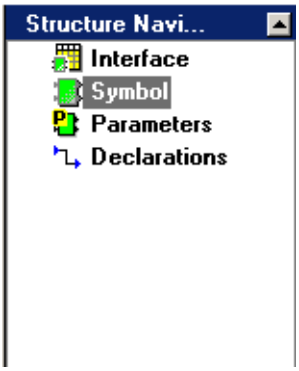
Procedure

1. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
2. Choose Graphical View from the Categories pane and specify the view type as: Interface from the File Types pane.
3. Set the language dialect as Verilog 2005.
4. Click Next to specify the new Verilog view location and name.
5. Click Finish. The Interface tabular IO editor is launched.

   Notice the additional **Signed** status column (for net or reg types) and **Value** column for outputs of type reg, integer, time.



6. You can click on the Symbol icon in the **Structure Navigator** to display the graphical view of the created interface.



Example: Creating a Verilog 2005 Interface

This example shows how a Verilog 2005 interface can be created and edited in HDS. It also shows how you can change the Verilog 2005 port display.

Prerequisites

Open the SCRATCH_LIB library of the Examples project.

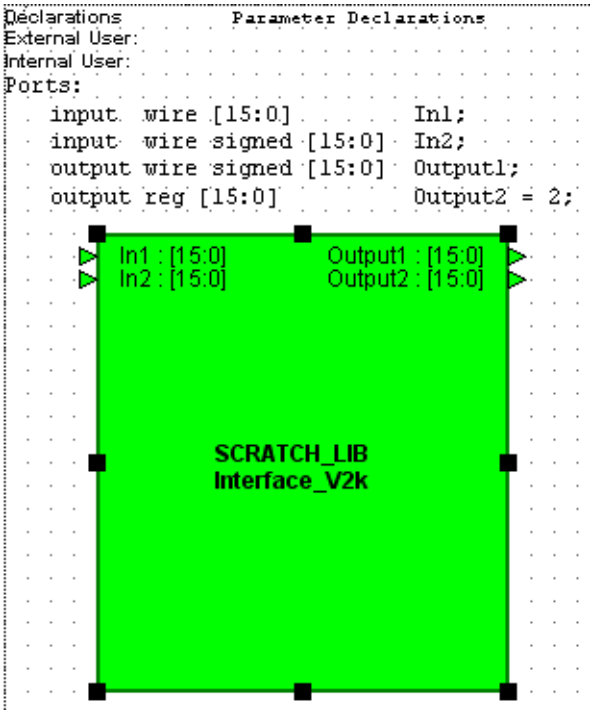
Steps

1. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
2. Choose Graphical View from the Categories pane and specify the view type as: Interface from the File Types pane.
3. Set the language dialect as Verilog 2005.
4. Click **Next** to display the **Specify View Name/Location** page. Select SCRATCH_LIB from the Library name dropdown list and enter Interface_V2k in the Design Unit entry field
5. Click Finish. The Interface tabular IO editor is launched.
6. Add the following signals and save the new interface view.

Table 2-1. Signals Table

Name	Mode	Signed	Type	Bounds	Value
In1	input		wire	[15:0]	
In2	input	signed	wire	[15:0]	
Out1	output	signed	wire	[15:0]	
Out2	output		reg	[15:0]	2

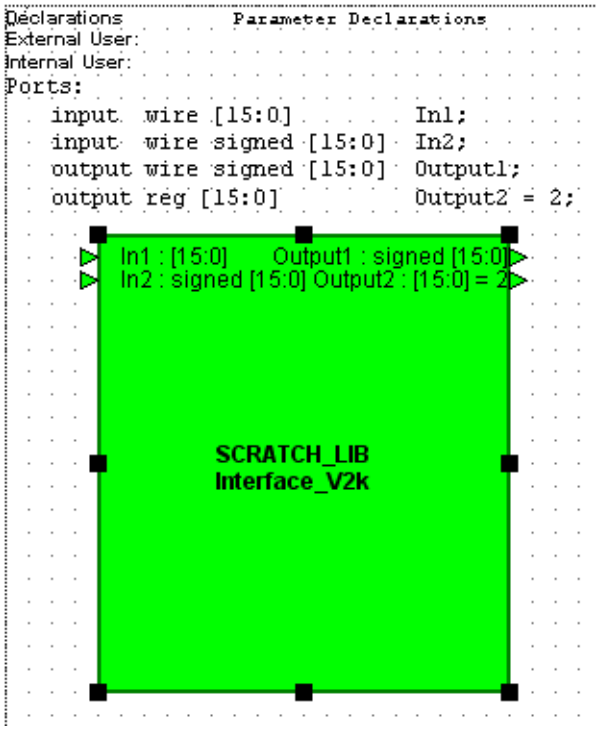
7. Click on the Symbol icon in the **Structure Navigator** to display the graphical view of the created interface Notice how the ports are declared using the default Ansi-C style.



8. Right click on the symbol to display the **Objects Properties** dialog box. The Symbol tab allows you update the symbol, symbol instances and Symbol/View update properties.

   You can change the display of symbol ports to show or hide the new Signed and Value properties.To change the display of the ports of the symbol to show itself click on the **Port Display** button in the Symbol group box. To change the display of the ports of the symbol instances click on the **Port Display** button in the.Symbol Instance group box.

9. Click on the Port Display button in the Symbol group box and set the Signed and Value options and save your changes. Notice how the port declarations on the symbol have been updated.



10. Close the symbol editor.

Note

Two way synchronization is supported between the Symbol tabular IO and the Signals table

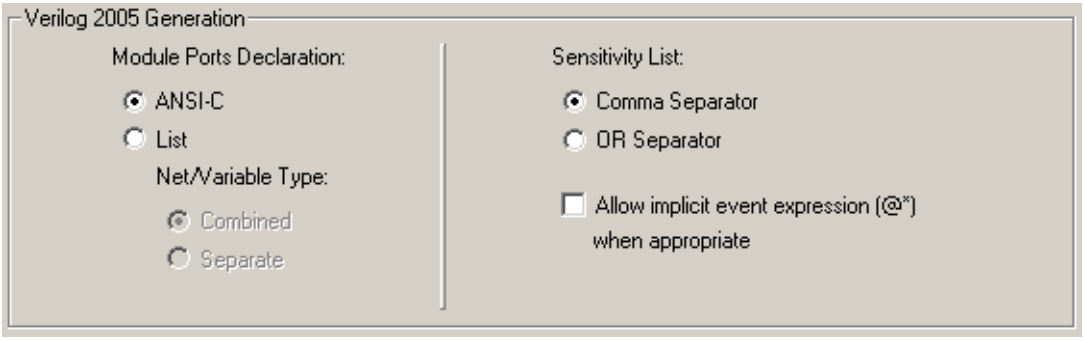
Creating a Text View for an Existing Verilog 2005 Interface

Verilog 2005 text views created based on a Verilog 2005 symbol contain a module interface that has port style generated according to the module ports declaration settings defined through the **Verilog Options** dialog.

Two way synchronization exists between the Verilog 2005 symbols and their corresponding views. Updates saved in one view are reflected in the other.

Procedure

1. Set the port style that you wish to have in your created text file by selecting **Options>Verilog** from the **Design Explorer** window to display the **Verilog Options** dialog.
2. On the **Style** tab set your port definition style preferences as ANSI-C or list.



3. Select a Verilog 2005 Interface design unit from the Design Unit browser.
4. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
5. Choose Verilog View from the Categories pane and specify the view type as: Module from the **File Types** pane.
6. Set the language dialect as Verilog 2005.
7. Click **Next** to specify the new Verilog view location and name.
8. Click **Finish**. The Designpad editor is launched showing the newly created Verilog File.

   The created Verilog file contains a module interface that has port style generated according to the module ports declaration settings defined through the **Verilog Options** dialog.

Related Topics

- Using the Design Content Creation Wizard
- Verilog 2005 Generation Preferences

Creating Verilog 2005 Views

HDS allows you to create both text and graphical Verilog 2005 views. Preferences set in the **Verilog Options** dialog are applied to the created views.

Creating Verilog 2005 Text Files

Verilog text files are created through the **Design Content Creation** wizard.

Procedure

1. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
2. Choose Verilog File from the **Categories** pane and Module or Include from the **File Types** pane.
3. Set the language dialect as Verilog 2005.

   If you choose to use a template define whether you want to specify your own or work with the default one.
4. Click Next to specify the new Verilog file location, name and format.
5. Click Finish. The **DesignPad** editor is launched displaying your newVerilog file.

Creating Verilog 2005 Graphical Views

Verilog 2005 graphical views are created through the **Design Content Creation** wizard.

Procedure:

1. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
2. Choose Graphical View from the **Categories** pane and specify the view type from the **File Types** pane.
3. Set the language dialect as Verilog 2005.
4. Click **Next** to specify the new Verilog view location and name.
5. Click **Finish**. The appropriate editor is launched displaying the selected graphical view.

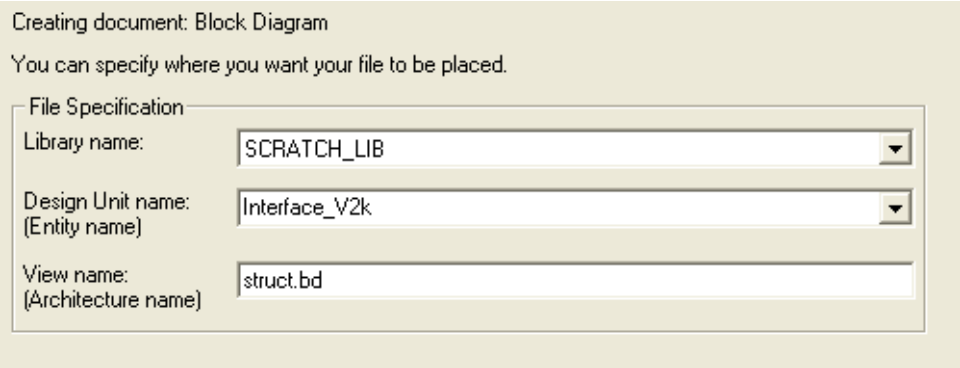
Example:Creating a Block diagram View

This is an example showing the creation of a block diagram view based on an existing V2k symbol. The block diagram view signal types, values and sign properties are edited to point the new supported V2k features.

Prerequisites

Select the Interface view Interface_V2K you created earlier. Refer to “Example: Creating a Verilog 2005 Interface” on page 33

1. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
2. Choose Graphical View from the **Categories** pane and select Block Diagram from the **File Types** pane.
3. Set the language dialect as Verilog 2005.
4. Click **Next** to display the **Specify View Name/Location** page. Accept the set values and click Next.



5. On the Specify Interface page the Symbol ports are listed.You can edit or update the displayed list.

	A	B	C	D
	Port Name	Mode	Type	Bounds
1	In1	input	wire	[15:0]
2	In2	input	wire	[15:0]
3	Output1	output	wire	[15:0]
4	Output2	output	reg	[15:0]
5				

6. Click **Finish** to launch the **Block Diagram** editor displaying the new BD view.
7. Select the output2 signal and right click to choose **Object Properties** from the popup menu. The Signals page of the **Object Properties** dialog is displayed.
8. Select wire from the Type field dropdown list. Notice that the Value field is disabled. Only output signals of type reg, integer and time can be given a value.Input signals can not have values.
9. Select the input2 signal and right click to choose **Object Properties** from the popup menu. The Signals page of the **Object Properties** dialog is displayed. Select real from the Type field dropdown list. Notice that the Signed check box is dimmed. Only signals of types net or reg can be signed.
10. Close your block diagram view.

Creating Mixed Dialect Designs

HDS allows you create mixed dialect designs: the same design unit can be presented as a Verilog 95, Verilog 2005 and a SystemVerilog view.

Some limitations exist on the creation of such mixed dialect designs based on the dialect and type of the default design view. The limitations can be summarized in the following table:

Table 2-2. Mixed Dialect Designs Limitations

Default View Type	Default View Dialect	Possibly Created Views
Text	Verilog 95	Verilog 95 graphical views
	Verilog 2005	Verilog 95 and Verilog 2005 graphical views.
	SystemVerilog	Verilog 95 and Verilog 2005 graphical views.
Graphical	Any Verilog dialect	Any text Verilog dialect view.

Note

Mixed dialect graphical views are not allowed.

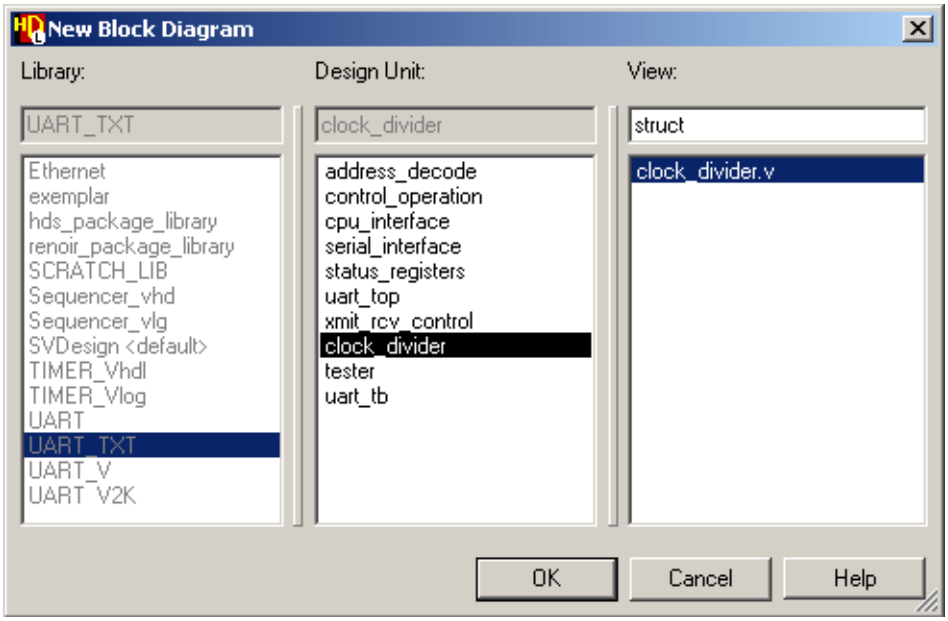
Procedure

1. Select a design unit from the Design Unit browser and do one of the following:
   - Open the **Design Content Creation** wizard by clicking the **New/Add** button in the shortcut bar.
   - Choose **New** from the popup menu.
2. Specify your design type and language following the rules stated in table Table 2-2 and complete the wizard steps.

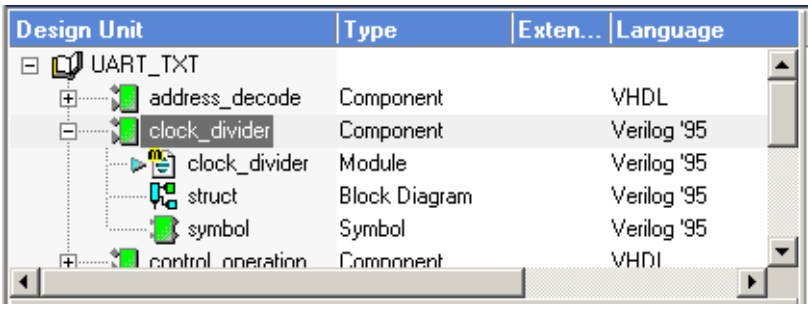
   A new view is created. A warning message is issued incase you try to create an invalid view.

Example

1. In the Design Explorer select the clock_divider design unit and choose **New>Graphical View>Block Diagram** from the popup menu. The New Block Diagram dialog is displayed.



2. Click ok to accept the defaults. HDS will create a Block Diagram view using the dialect of the default view.



Importing Verilog 2005 Designs

Existing Verilog 2005 designs can be added to HDS projects through the **Add Existing Design** wizard. When adding a design to a HDS project you can choose to move a copy of the design to the HDS projects' directory or point to it in it's current location.

Procedure

1. Start the **Add Existing Design** Wizard by choosing **File>Add>Existing Design**:

a. Specify the method to add design content by choosing **Copy Specified Files or Point to Specified files** from the Add Method group box.

b. Specify whether you would like to use a filelist to obtain information on design content by setting the Use Filelist option.Specify the files to add.

2. Specify the method to determine the Verilog dialect used as auto or Specified.

Table 2-3. Verilog Dialect Setting Options

Option	Description
Auto	The verilog dialect used is automatically detected. A single design may include different verilog dialects
Specified	The user can mark all design files as verilog 2005 even if they do not contain any verilog 2005 constructs.

3. Complete the **Add Existing Design** wizard steps. The **Design Explorer** window is displayed showing the added design.

Related Topics

- Adding Existing Design Content to a Project
- Adding Design Files to a Library
- Removing Design Content
- Add Existing Design Wizard

Editing Verilog 2005 Signals

Verilog 2005 signals or ports are described by the following 7 fields:Name, Mode, Signed, Type, Bounds and Value.

The values of the above mentioned fields can be edited in the Object Properties dialog, the Signals table, the Interface table or Symbol. In the case of a block diagram view, inplace editing is possible as well.

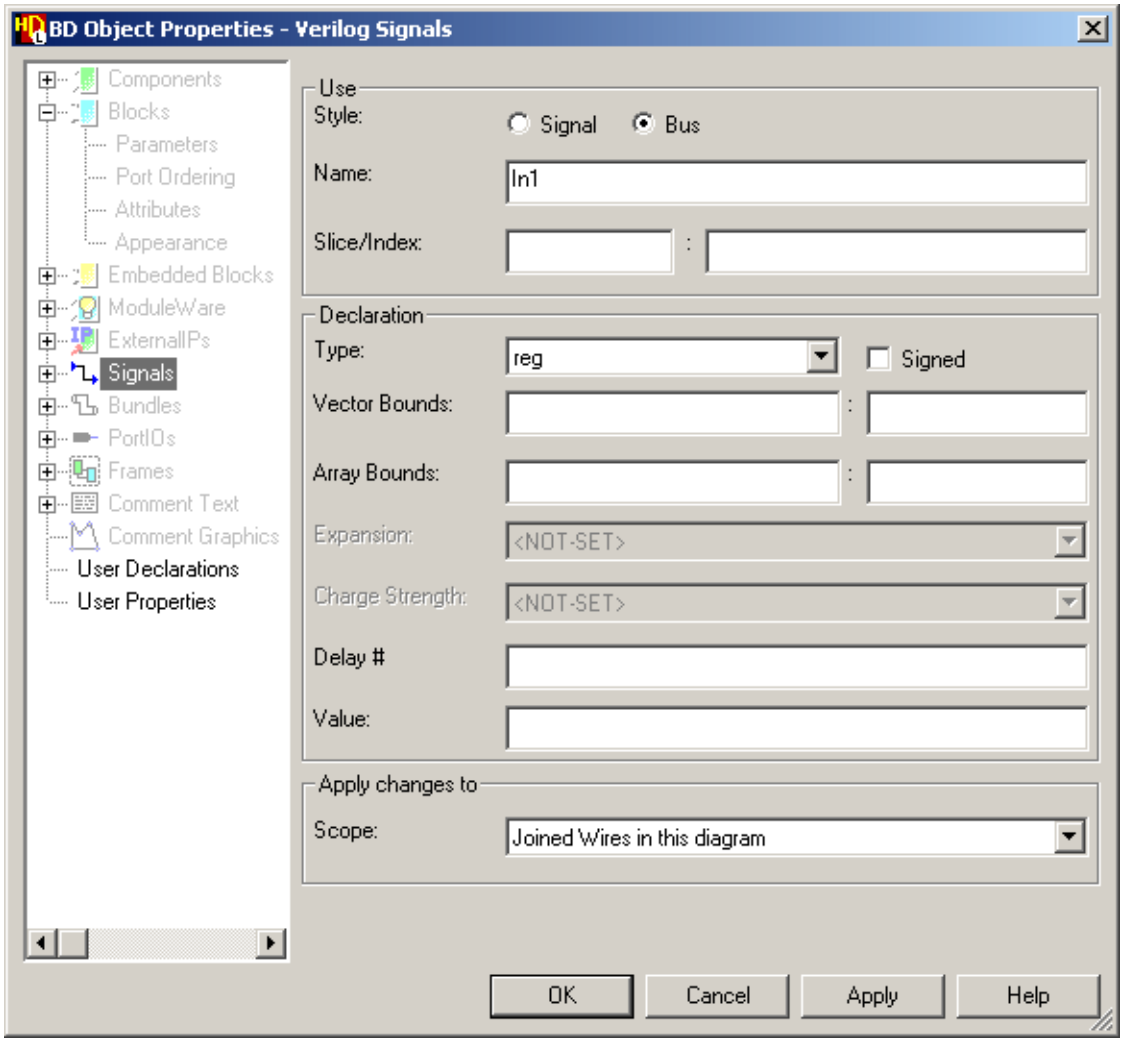
Procedure

The following steps describe the editing of Verilog 2005 signals using the **Object Properties** dialog.

1. Open a Verilog 2005 Block Diagram view and double click on any of it’s signals. The Object Properties dialog is displayed. Set the properties as for any Verilog or VHDL signal

2. For all types other than Time and integer specify the signed property.

3. You may specify a value for any of your output signals.Notice that any changes are synchronized with the Signals table and with the view declaration text.



Related Topics

- Editing Object Properties

Generating HDL from Verilog 2005 Graphical Views

You can use HDS to generate HDL code for your graphical designs. HDS allows you to have a degree of control over the style of the Verilog code generated through the **Verilog Options** dialog box. It gives you even more control over the code generated from Verilog 2005 graphical views, this can be useful in the case of downstream tools with limited Verilog 2001/2005 support.

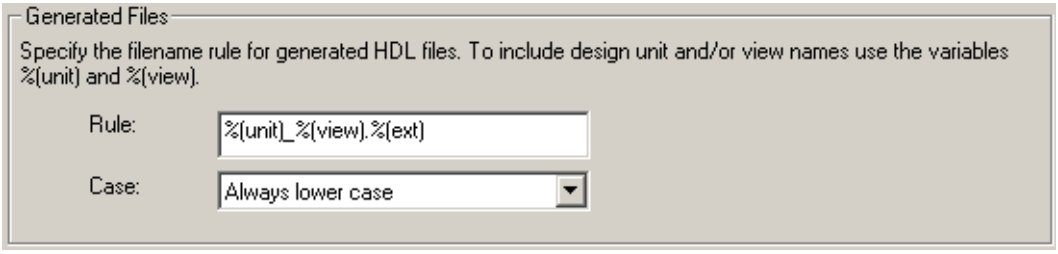
Note

Moduleware instances in Verilog 2005 designs follow the Verilog 2005 code generation settings.

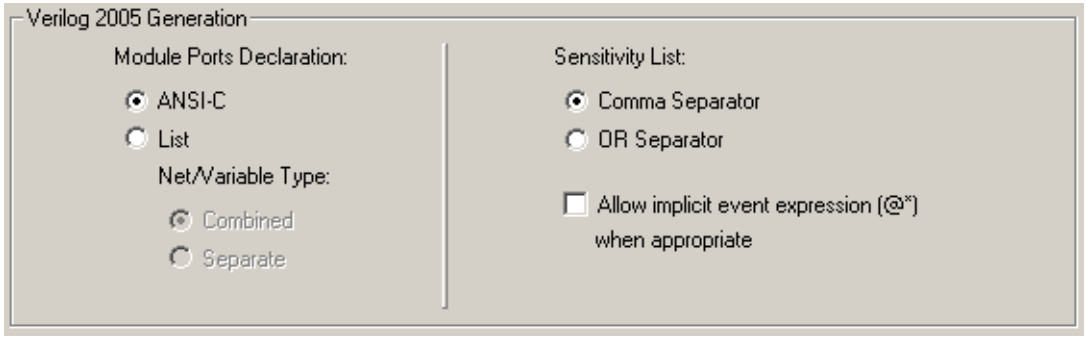
Procedure

1. In the **Design Explorer** window select **Options>Verilog** to display the **Verilog Options** dialog.This is where you will be setting all your code generation preferences.

   a. In the Generated Files Group box of the **Files** tab set the file naming rules you wish to use when generating Verilog files or leave the default settings.

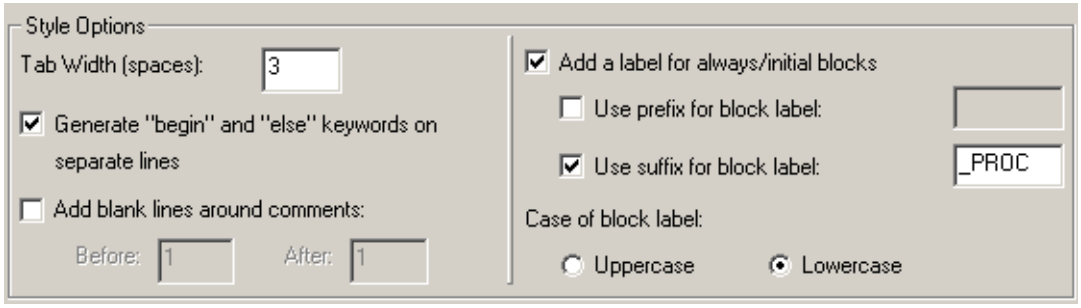


   b. In the Verilog 2005 Generation Group box specify some Verilog 2005 code style features.



   - Specify whether you want to generate ports in a port list or using the ANSII-C style. On choosing to generate listed ports the list sub-options allow you to combine your generated ports' data types with the port declaration or have them in separate statements.
   - For leaf level views you can specify the sensitivity list separator style to be a comma or an "OR" . You can also choose to allow implicit event expression @* where appropriate.

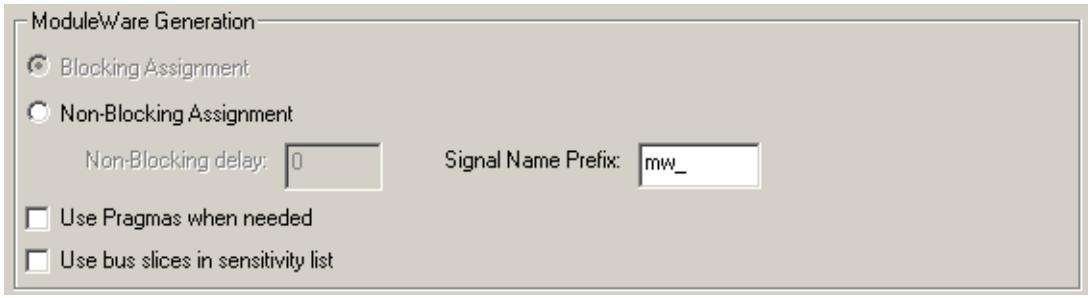
c. Specify the style features of the generated code in the Style Options Group Box of the Style tab or choose to leave the defaults



You can

- Specify the default tab width setting for Verilog text views by entering the number of spaces to indent.
- Choose to generate HDL with the begin and else keywords on separate lines.
- Add blank lines around comments by specifying the number of lines to add before and after the comment line.
- Choose to add a label to generated always or initial code blocks and specify a prefix or suffix which is added to the label names. You can also choose whether to use uppercase or lowercase labels.

d. If your design includes ModuleWare components you can specify the style features of the generated code.



In the ModuleWare Generation Group Box of the Style tab you can do the following:.

- Choose blocking assignments or non-blocking assignments to registers in the HDL for sequential (clocked) ModuleWare components.
- Optionally enter a non-blocking delay when non-blocking assignment is set. (Note that the default value 0 means no delay.)
- Set the signal name prefix used for ModuleWare signals in the generated HDL
- Choose to use Pragmas when needed.

- Choose to use bus slices in sensitivity lists depending on whether your simulation tool support slices or not; If it doesn't do not set this option as this will lead to including only the bus name in the sensitivity list on generation.

e. You can specify parameter generation properties. In the **Parameters** Group Box of the Style tab you can do the following:

- Choose whether all Verilog parameters defined in the symbol are automatically initialized with default values and displayed when a component is instantiated. If this option is not set, only the overridden parameters are displayed.
- Choose whether a synopsys template pragma is added above the parameter declaration in the generated HDL for the symbol. (This option is enabled for backward compatibility but should normally be unset if you are not using the Synopsys synthesis tools.)

2. In the **Design Explorer** window select a Verilog 2005 design unit and choose a generation option from the **Tasks>Generation** submenu. Check the log window for any warnings and errors.
3. To view your generated file choose **HDL> View Generated HDL.**

Related Topics

- Preferences for HDL Views

Chapter 3
VHDL Support

Introduction

HDS provides both text-level and graphical support for the VHDL language. In this chapter, we will provide an overview of the features in HDS designed to support the VHDL 2008 language.

Importing existing VHDL 2008 designs in addition to creating new designs is now possible.

VHDL text designs can be instantiated inside Block Diagram and IBD views as well as visualized or converted to graphics.

Note

HDS supports the following VHDL 2008 constructs: external names, force and release, read out ports, signal expressions in portmaps, Breakpoints and TextIO.

VHDL 2008 New Language Features

- Breakpoints:
  - A new package was added to the “std” library in HDS. The package name is “ENV”. This package has “**stop**” and “**finish**” procedures necessary for supporting breakpoints.
- Improved IO enhancements:

- The package "**TEXTIO**" in HDS **ieee** library was updated (Header and Body). Added support to unary **logical reduction operators** (used in TEXTIO body)
- Removed three packages (Header and Body) from HDS **ieee** library and added their new versions internally in HDS VHDL parser (for copyright reasons). The three packages are: *std_logic_1164*, *numeric_std* and *numeric_bit*.
- Added three new VHDL 2008 packages internally in HDS VHDL parser: *fixed_pkg*, *float_pkg* and *fixed_float_types*.
- Added predefined and implicit "to_string" functions, and predefined aliases required for Improved IO enhancements.

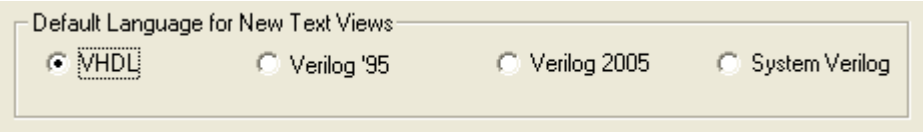
- Miscellaneous:
  - Updated standard package in HDS **std** library with a newer version.
  - You can use the old set of packages by replacing the following directory with the one having the old packages: <HDS home dir>/hdl_libd/ieee/hdl

Setting VHDL as the Default HDS Language

HDS allows you to set VHDL as the default language for all newly created designs. This can be helpful if VHDL is your most frequently used design language.

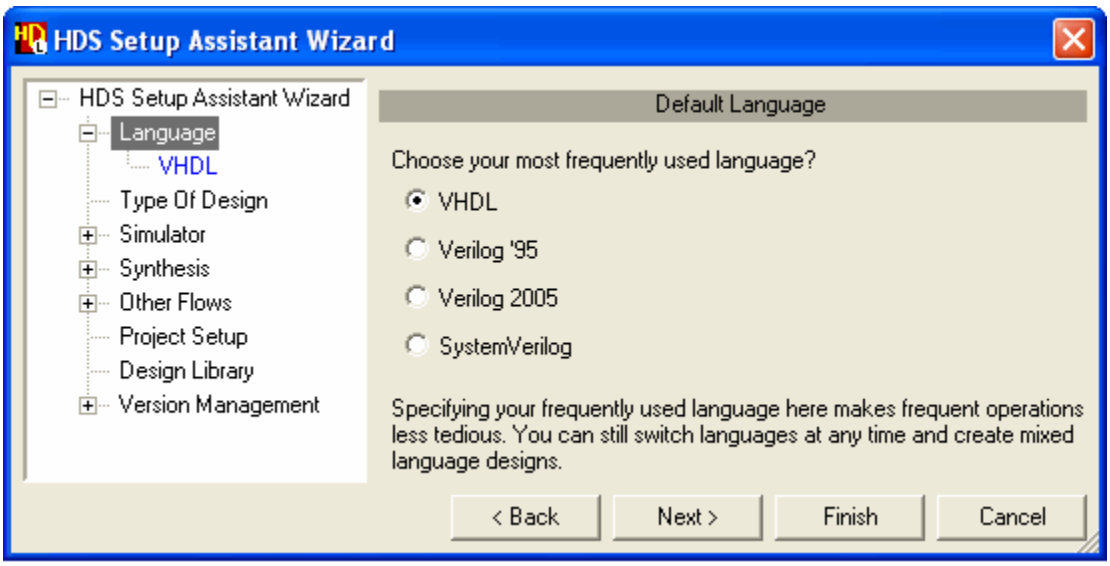
Procedure

1. Choose **Options> Main** from the Design Explorer window. The **Main Settings** dialog appears.
2. On the **General** tab, in the Default Language for New Views group box, choose the VHDL option then click OK.



Tip: You can change the set default language for a specific view from the Design Content Creation Wizard.

Also, you can set VHDL as the default language by choosing **Help> HDS Setup Assistant**. The **HDS Setup Assistant Wizard** appears. Select the Language node and set the VHDL option in the left pane of the window. Click **Finish** to save your new setting.



Related Topics

- HDS Setup Assistant Wizard
- Main Settings Dialog

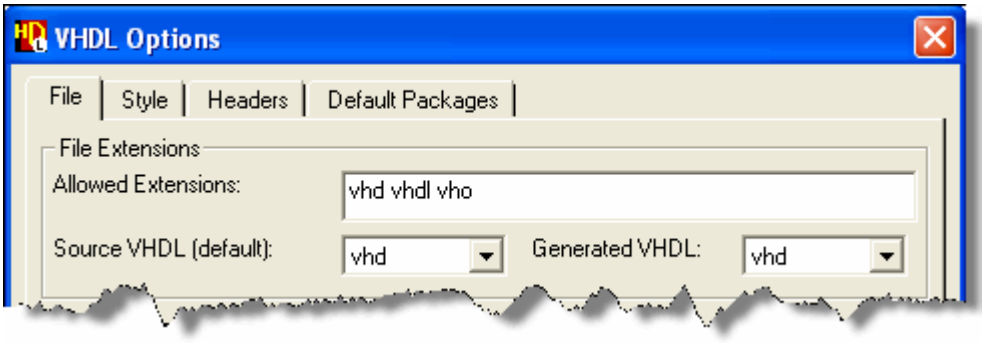
Setting VHDL Language Preferences

New VHDL designs are created according to a set of predefined options. The options are related to file naming, file headers and default packages and are set through the VHDL Options dialog box.

Procedure

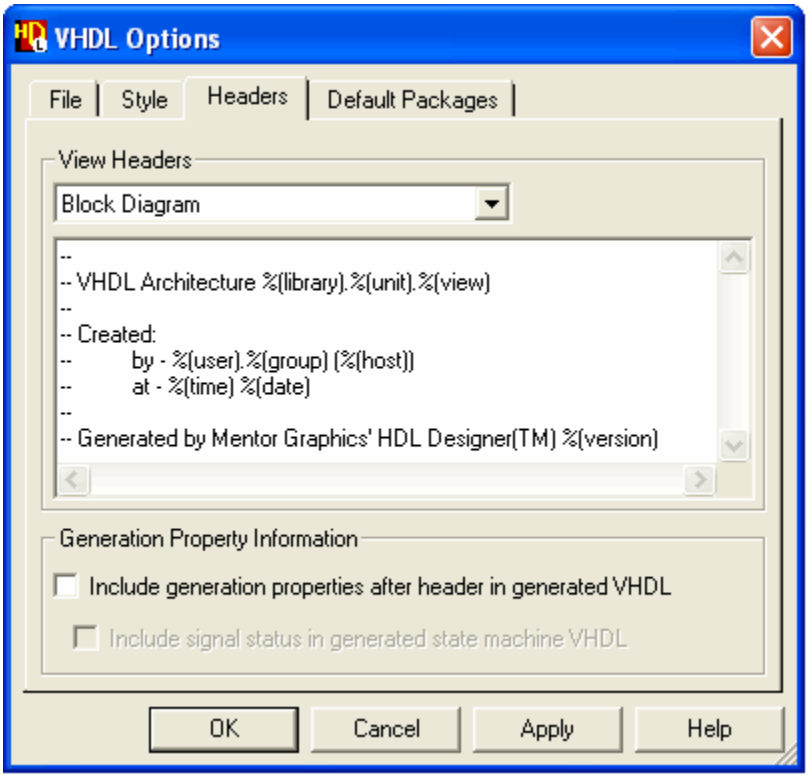
1. In the **Design Explorer** window, select **Options> VHDL** to display the **VHDL Options** dialog box.

2. On the **File** tab, specify your default VHDL file extension by choosing from the Source VHDL (default) dropdown list.



3. On the **Headers** tab, choose from a pull-down list of view headers. The selected header is used in the created view.

   You can also choose to include the generation properties after the header in the generated VHDL. If this option is set, you can also include the signal status in generated state machine VHDL.



4. Click **OK**. The above stated preferences will be added to any new view.

Working with VHDL Interfaces

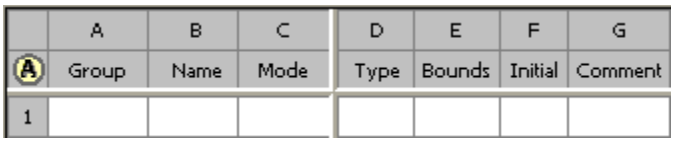
An HDS interface defines the output and input ports for a design. The interface is presented in either a tabular form and is referred to as an Interface or a graphical form and is referred to as a Symbol.

Creating a VHDL Interface

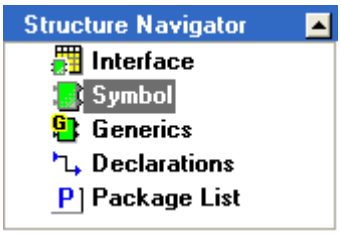
Component interfaces are created through the **Design Content Creation Wizard** and edited through the tabular IO and symbol editors.

Procedure

1. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
2. Choose “Graphical View” from the Categories pane and specify the view type as “Interface” from the File Types pane.
3. Set the language as VHDL.
4. Click Next to specify the new VHDL view location and name.
5. Click Finish. The Interface tabular IO editor is launched.



6. You can click on the Symbol icon in the **Structure Navigator** to display the graphical view of the created interface.



Example: Creating a VHDL Interface

This example shows how a VHDL interface can be created and edited in HDS. It also shows how you can change the VHDL port display.

Prerequisites

Open the SCRATCH_LIB library of the Examples project.

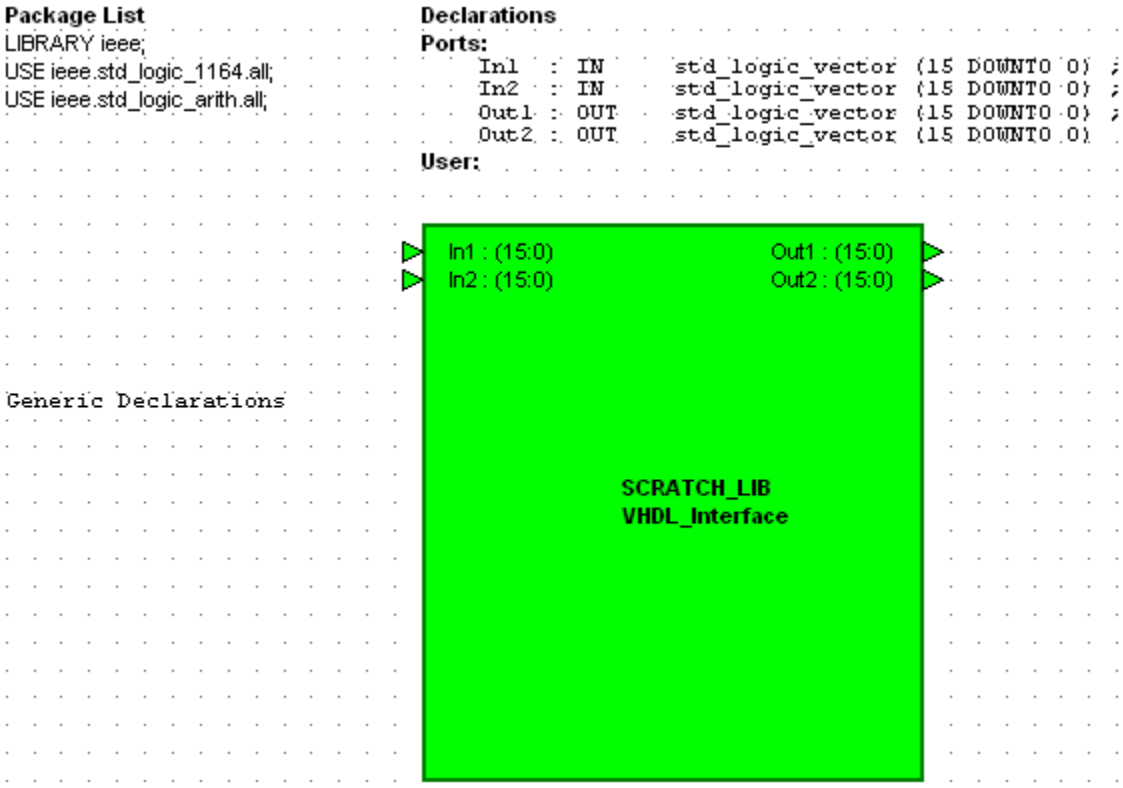
Steps

1. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
2. Choose “Graphical View” from the Categories pane and specify the view type as “Interface” from the File Types pane.
3. Set the language as VHDL.
4. Click **Next** to display the **Specify View Name/Location** page. Select SCRATCH_LIB from the Library name dropdown list and enter VHDL_Interface in the Design Unit name field.
5. Click **Finish**. The Interface tabular IO editor is launched.
6. Add the following signals and save the new interface view.

Table 3-1. Signals Table

Name	Mode	Type	Bounds
In1	IN	std_logic_vector	(15:0)
In2	IN	std_logic_vector	(15:0)
Out1	OUT	std_logic_vector	(15:0)
Out2	OUT	std_logic_vector	(15:0)

7. Click on the Symbol icon in the **Structure Navigator** to display the graphical view of the created interface. Notice how the ports are declared using the default Ansi-C style.



8. Right click on the symbol and choose **Objects Properties** to display the **Symbol/Interface Properties** dialog box. The Symbol tab allows you update the symbol and symbol instance options.

   To change the display of the ports of the symbol to show itself click on the **Port Display** button in the Symbol group box. To change the display of the ports of the symbol instances click on the **Port Display** button in the Symbol Instance Options group box.

9. Click on the Port Display button in the Symbol group box, the **Port Display Control** dialog box is displayed.

10. In the Fields to display group box, update the format and visibility of usage and declaration information attached to all ports then save your changes. Notice how the port declarations on the symbol have been updated.

11. Close the symbol editor.

Creating a Text View for an Existing VHDL Interface

HDS allows you to create a text view for an existing VHDL interface. Preferences set in the **VHDL Options** dialog are applied to the created views.

Procedure

1. Set the preferences that you wish to have in your created text file by selecting **Options**> **VHDL** from the **Design Explorer** window to display the **VHDL Options** dialog.
2. Select a VHDL Interface design unit from the Design Unit browser.
3. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
4. Choose “VHDL File” from the Categories pane and specify the view type as “Entity” from the **File Types** pane.
5. Set the language as VHDL 2008.
6. Click **Next** to specify the new VHDL view location and name.
7. Click **Finish**. The Designpad editor is launched showing the newly created VHDL File.

Related Topics

- Using the Design Content Creation Wizard

Creating VHDL 2008 Views

HDS allows you to create text VHDL 2008 view. Preferences set in the **VHDL Options** dialog are applied to the created views.

Creating VHDL 2008 Text Files

VHDL text files are created through the **Design Content Creation Wizard**.

Procedure

1. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
2. Choose “VHDL File” from the **Categories** pane and “Entity” or “Architecture” from the **File Types** pane.
3. If you choose to use a template, define whether you want to specify your own or work with the default one.
4. Set the language as VHDL 2008.
5. Click Next to specify the new VHDL file location, name and format.
6. Click Finish. The **DesignPad** editor is launched displaying your new VHDL file.

Creating VHDL Graphical Views

VHDL graphical views are created through the **Design Content Creation** wizard.

Procedure:

1. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
2. Choose “Graphical View” from the **Categories** pane and specify the view type from the **File Types** pane.
3. Set the language as VHDL.
4. Click **Next** to specify the new VHDL view location and name.
5. Click **Finish**. The appropriate editor is launched displaying the selected graphical view.

Example: Creating a Block Diagram View

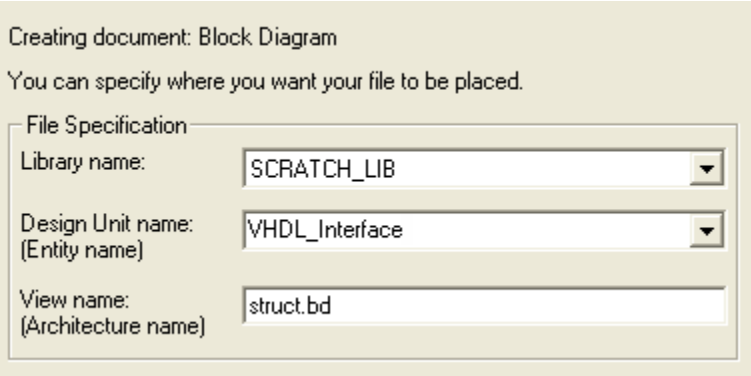
This is an example showing the creation of a block diagram view based on an existing VHDL_Interface symbol.

Prerequisites

Select the Interface view VHDL_Interface you created earlier. Refer to "Example: Creating a VHDL Interface" on page 51.

Steps

1. Invoke the **Design Content Creation Wizard** by clicking the **New/Add** button in the **Design Explorer** shortcut bar.
2. Choose "Graphical View" from the **Categories** pane and select "Block Diagram" from the **File Types** pane.
3. Set the language as VHDL.
4. Click **Next** to display the **Specify View Name/Location** page. Accept the set values then click Next.



5. On the **Specify Interface** page, the Symbol ports are listed. You can edit or update the displayed list.

	A	B	C	D
	Port Name	Mode	Type	Bounds
1	In1	IN	std_logic_vector	(15 DOWNTO 0)
2	In2	IN	std_logic_vector	(15 DOWNTO 0)
3	Out1	OUT	std_logic_vector	(15 DOWNTO 0)
4	Out2	OUT	std_logic_vector	(15 DOWNTO 0)
5				

6. Click **Finish** to launch the **Block Diagram** editor displaying the new BD view.

7. Select the Out2 signal, then right click and choose **Object Properties** from the popup menu. The Signals page of the **Object Properties** dialog is displayed.
8. Select “std_logic” from the Type field dropdown list. Notice that the Bounds field is disabled.
9. Close your block diagram view.

Importing VHDL 2008 Designs

Existing VHDL 2008 designs can be added to HDS projects through the **Add Existing Design** wizard. When adding a design to a HDS project, you can choose to move a copy of the design to the HDS projects’ directory or point to it in it’s current location.

Procedure

1. Start the **Add Existing Design** dialog by choosing **File> Add> Existing Design**.
   a. Specify the method to add design content by choosing **Point to Specified Files** or **Copy Specified Files**.
   b. Specify whether you would like to use a filelist to obtain information on design content by setting the Use an Existing Filelist option. Specify the files to add.
2. Complete the **Add Existing Design** wizard steps. The **Design Explorer** window is displayed showing the added design.

Related Topics

- Adding Existing Design Content to a Project
- Adding Design Files to a Library
- Removing Design Content
- Add Existing Design Wizard

Editing VHDL Signals

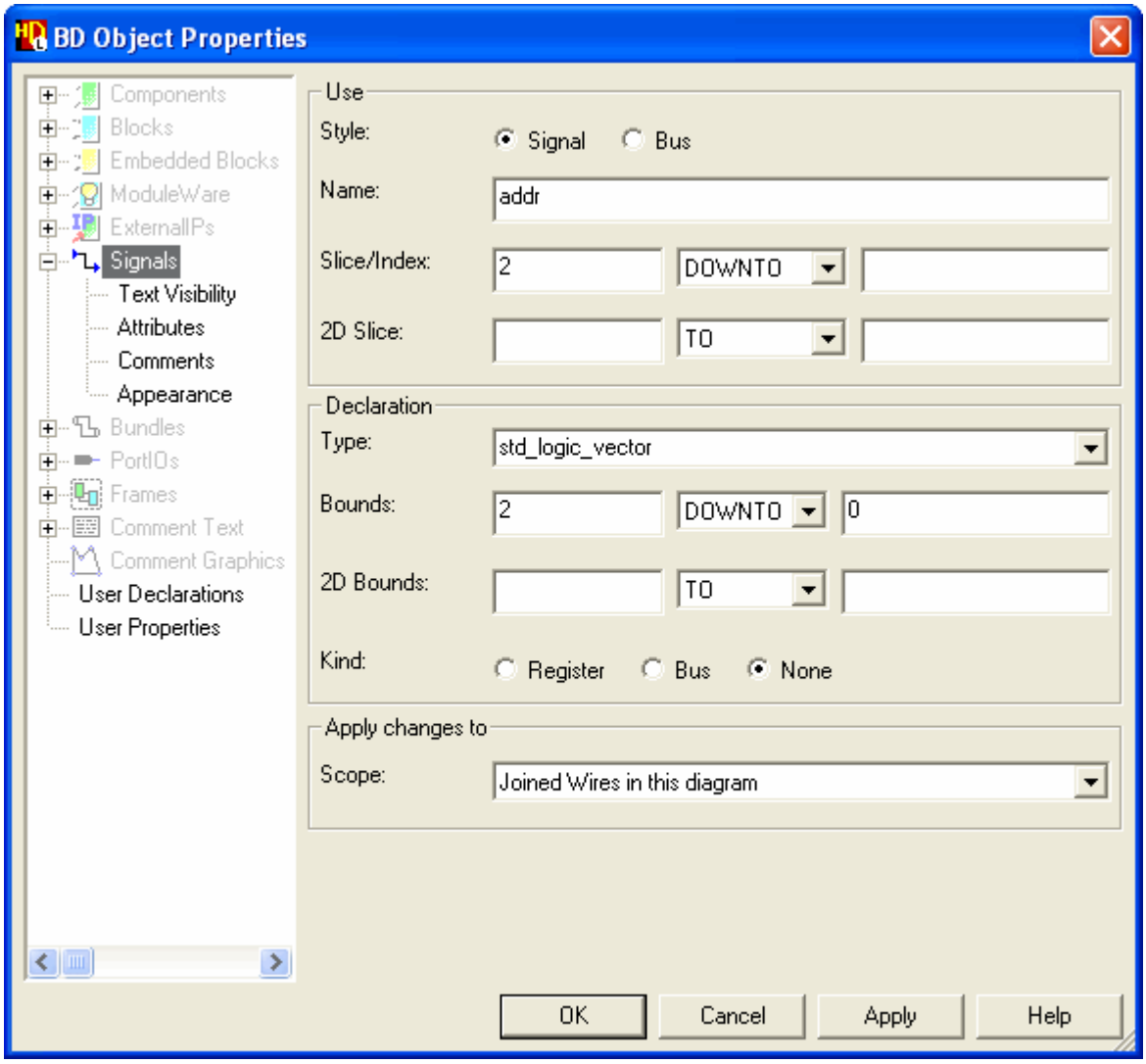
VHDL signals or ports are described by the following fields: Name, Mode, Type and Bounds.

The values of the above mentioned fields can be edited in the Object Properties dialog, the Signals table, the Interface table or Symbol. In the case of a block diagram view, inplace editing is possible as well.

Procedure

The following steps describe the editing of VHDL signals using the **Object Properties** dialog.

1. Open a VHDL Block Diagram view and double click on any of it's signals. The Object Properties dialog is displayed. Set the properties as for any Verilog or VHDL signal.
2. You may specify a value for any of your output signals. Notice that any changes are synchronized with the Signals table and with the view declaration text.



Related Topics

- Editing Object Properties

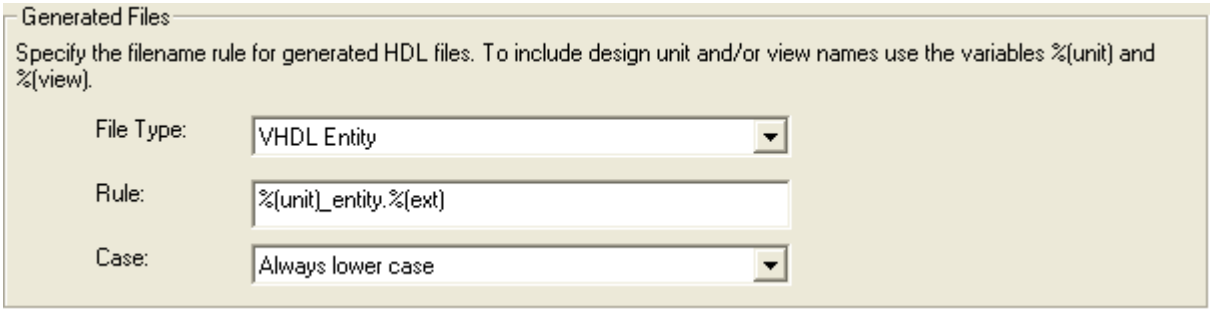
Generating HDL from VHDL Graphical Views

You can use HDS to generate HDL code for your graphical designs. HDS allows you to have a degree of control over the style of the VHDL code generated through the **VHDL Options** dialog box.

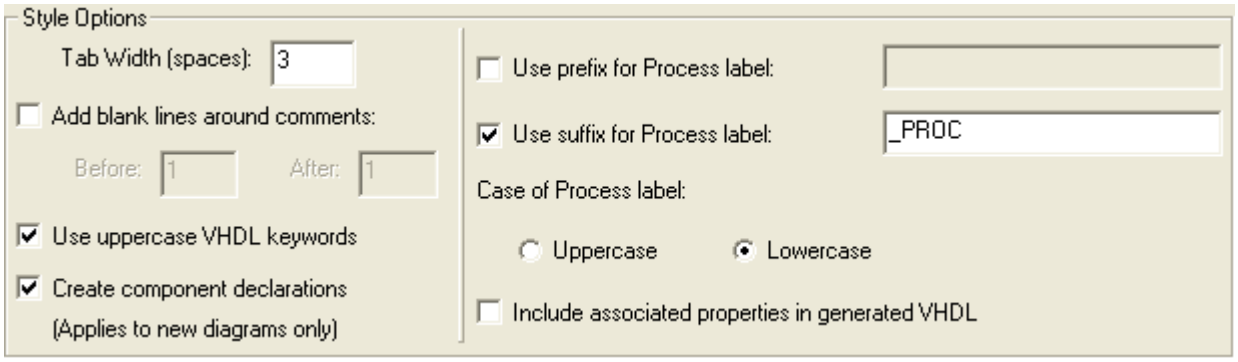
Procedure

1. In the **Design Explorer** window select **Options> VHDL** to display the **VHDL Options** dialog.This is where you will be setting all your code generation preferences.

   a. In the Generated Files group box of the **File** tab, set the file naming rules you wish to use when generating VHDL files or leave the default settings.



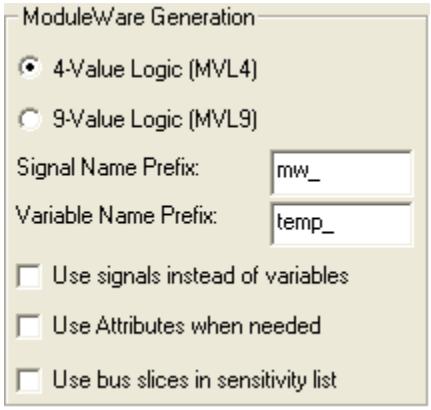
   b. Specify the style features of the generated code in the Style Options group box of the **Style** tab or choose to leave the defaults



   You can

   - Specify the default tab width setting for VHDL text views by entering the number of spaces to indent.
   - Add blank lines around comments by specifying the number of lines to add before and after the comment line.
   - Choose to use uppercase VHDL keywords.
   - Create component declarations. Note that they will be applied to new diagrams only.
   - Choose to specify a prefix or suffix which is added to the label names. You can also choose whether to use uppercase or lowercase labels.
   - Choose to include associated properties in generated VHDL.

c. If your design includes ModuleWare components, you can specify the style features of the generated code.



In the ModuleWare Generation group box of the **Style** tab, you can do the following:

- Choose 4-Value Logic (MVL4) to generate ModuleWare components using the std_ulogic values 0, 1, X and Z only or 9-Value Logic (MVL9) to allow all the std_logic or std_ulogic values 0, 1, L, H, X, -, U, W and Z. (This option controls whether the values in conditions and CASE selects are compared using MVL4 or MVL9 values.)
- Set the Signal/Variable Name Prefix used for ModuleWare signals in the generated HDL.
- Choose to use signal names instead of internal variables in the VHDL generated for a ModuleWare part.
- Choose to use Attributes when needed.
- Choose to use bus slices in sensitivity lists depending on whether your simulation tool support slices or not; If it doesn't, do not set this option as this will lead to including only the bus name in the sensitivity list on generation.

d. In the Test Bench Generation group box of the **Style** tab, you can specify the default names for the test bench and the testers views.

2. In the **Design Explorer** window, select a VHDL design unit and choose a generation option from the **Tasks> Generate** submenu. Check the log window for any warnings and errors.
3. To view your generated file, choose **HDL> View Generated HDL.**

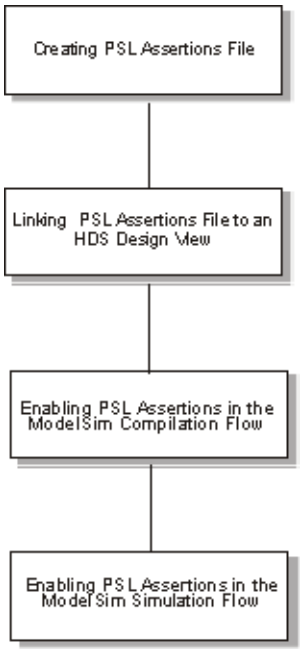
Related Topics

- Preferences for HDL Views

Chapter 4
PSL Assertions

Introduction

PSL is a language for the formal specification of hardware. PSL files contain PSL verification units(vunits).HDS supports the creation of PSL assertion files as side data files linked to a design view and recognized by the ModelSim flow.



Creating PSL Assertion Files

PSL assertion files are created through the Design Content Creation wizard. You can choose to place your PSL files in the Design Units pane, the Design Files pane or the Side Data pane of the Design Explorer Subwindow.

Note

Files created in the SideData pane are automatically associated to the specified design view.

To create a PSL file

1. Invoke the **Design Content Creation Wizard**.

2. Choose Registered view from the **Categories** pane and Property Specification Language from the **File Types** pane.
3. To base your assertions file on a predefined template set the Use Templates option otherwise click Next to start with a blank file.

   If you choose to use a template define whether you want to specify your own or work with the default one.
4. Click Next to specify the new PSL file location, name and format. At this stage you can do one of the following
   - Create your PSL file as a stand-alone file not linked to any design view by selecting Design Unit from the Section dropdown list.
   - Link your PSL file to a specific design view by choosing to place your files in the Sidedata or Design Files section.Refer to "Linking PSL Assertions to HDS Design Views" on page 62.
5. Click Finish. The DesignPad editor is launched displaying your new PSL file.

Linking PSL Assertions to HDS Design Views

For PSL assertion files to be recognized during ModelSim compilation and/or simulation the following must be satisfied:

- The files must be attached to an HDS design view.
- The files must be set as Active.

Tip: PSL files linked to a design view appear in the SideData pane.HDS treats these linked files as sidedata files to the specified design view.

Attaching PSL files to a Design View

PSL files can be linked to a design view at the time of their creation. Another option is to create or import a PSL file and later link it to any of your design views.

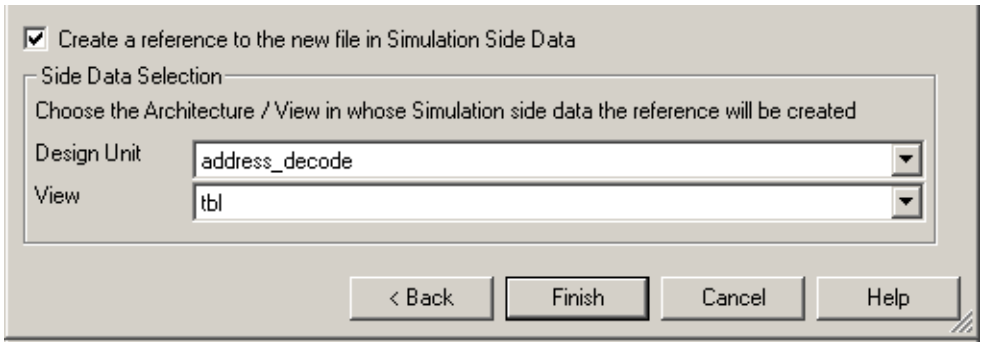
To attach a PSL file to a design view at the time of it's creation:

1. Refer to "Creating PSL Assertion Files" on page 61 and follow steps 1, 2 and 3.
2. On the Specify View Name/Location page of the Design Content Creation wizard do one of the following:
   - Choose SideData files from the Section dropdown list. Complete the File Specification and Side Data sections.

You now have a PSL file stored as a side data file for the design view you specified.

- Choose Design Files from the Section dropdown list and complete yor file specification section. You have the option to link this file to a design view by setting the **Create a reference to the new file in Simulation Side Data** option. If you do so you then have to choose the design view in which the reference will be created.

  You now have a PSL file stored as a design file and a reference to that PSL file stored in it's simulation side data.



To attach a PSL file to an existing design view:

1. In the Design Explorer window select the design view to which you wish to link a PSL file.
2. Choose Properties from the popup cascade menu. This will display the design view Properties dialog.
3. Click the ADD button on the Simulation page to launch the Import File dialog.
4. Browse for your PSL file. Choose whether to import or reference it in its current location and click OK.

   The PSL file or a reference to it appears in the sidedata files of the specified design view.

To detach a PSL file from a design view:

1. In the Design Explorer window select the design view from which you want to unlink a PSL file.
2. Choose Properties from the popup cascade menu. This will display the design view Properties dialog.
3. Select the file you want to unlink and click the Remove button on the Simulation page. Notice that the PSL file is removed from the design view's side data files.

Activating PSL Files

PSL files are created as active views i.e they are automatically detected during ModelSim compilation and simulation. If you would like to set them inactive do the following:

1. In the Design Explorer window select the design view whose PSL files are to be set inactive.
2. Choose Properties from the popup cascade menu. This will display the design view Properties dialog.
3. Select a PSL file and click the **Unset Active** button on the Simulation page.

You can also select your PSL file in the Side Data pane and from the popup cascade menu choose set active or inactive.

Including PSL Files in ModelSim Compilation/ Simulation Flows

ModelSim can detect the presence of active PSL files and use them in design compilation and simulation flows.

To include PSL files in ModelSim compilation/simulation flows:

1. Open the Tasks/Templates browser in the design explorer.
2. Select ModelSim Compile/ModelSim Simulate and choose settings from the popup cascade menu to display the ModelSim Compile Settings/ModelSim Simulate Settings dialog.
3. Select the General tab of the ModelSim Compile dialog/the Design tab of the ModelSim Simulate dialog.
4. Make sure that the Exclude PSL(-nopsl) is not set. If set a **-nopsl** switch is passed to the ModelSim compiler /simulator all linked assertions are ignored.

End-User License Agreement

The latest version of the End-User License Agreement is available on-line at:
www.mentor.com/eula

IMPORTANT INFORMATION

USE OF ALL SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF SOFTWARE INDICATES CUSTOMER'S COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.

END-USER LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Software (as defined in Section 2) and hardware (collectively "Products") between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If Customer does not agree to these terms and conditions, promptly return or, in the case of Software received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

1. **ORDERS, FEES AND PAYMENT.**

   1.1. To the extent Customer (or if agreed by Mentor Graphics, Customer's appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement ("Order(s)"), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement, any applicable addenda and the applicable quotation, whether or not these documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order will not be effective unless agreed in writing by an authorized representative of Customer and Mentor Graphics.

   1.2. Amounts invoiced will be paid, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice(s). Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer's sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer's behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.

   1.3. All Products are delivered FCA factory (Incoterms 2000), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics retains a security interest in all Products delivered under this Agreement, to secure payment of the purchase price of such Products, and Customer agrees to sign any documents that Mentor Graphics determines to be necessary or convenient for use in filing or perfecting such security interest. Mentor Graphics' delivery of Software by electronic means is subject to Customer's provision of both a primary and an alternate e-mail address.

2. **GRANT OF LICENSE.** The software installed, downloaded, or otherwise acquired by Customer under this Agreement, including any updates, modifications, revisions, copies, documentation and design data ("Software") are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors, who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form (except as provided in Subsection 5.2); (b) for Customer's internal business purposes; (c) for the term of the license; and (d) on the computer hardware and at the site authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Customer may have Software temporarily used by an employee for telecommuting purposes from locations other than a Customer office, such as the employee's residence, an airport or hotel, provided that such employee's primary place of employment is the site where the Software is authorized for use. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or services purchased, apply to the following: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be technically implemented through the use of authorization codes or similar devices); and (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions. For the avoidance of doubt, if Customer requests any change or enhancement to Software, whether in the course of

receiving support or consulting services, evaluating Software, performing beta testing or otherwise, any inventions, product improvements, modifications or developments made by Mentor Graphics (at Mentor Graphics' sole discretion) will be the exclusive property of Mentor Graphics.

3. **ESC SOFTWARE.** If Customer purchases a license to use development or prototyping tools of Mentor Graphics' Embedded Software Channel ("ESC"), Mentor Graphics grants to Customer a nontransferable, nonexclusive license to reproduce and distribute executable files created using ESC compilers, including the ESC run-time libraries distributed with ESC C and C++ compiler Software that are linked into a composite program as an integral part of Customer's compiled computer program, provided that Customer distributes these files only in conjunction with Customer's compiled computer program. Mentor Graphics does NOT grant Customer any right to duplicate, incorporate or embed copies of Mentor Graphics' real-time operating systems or other embedded software products into Customer's products or applications without first signing or otherwise agreeing to a separate agreement with Mentor Graphics for such purpose.

4. **BETA CODE.**

4.1. Portions or all of certain Software may contain code for experimental testing and evaluation ("Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to Customer a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. This grant and Customer's use of the Beta Code shall not be construed as marketing or offering to sell a license to the Beta Code, which Mentor Graphics may choose not to release commercially in any form.

4.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer's use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer's evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

4.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on Customer's feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this Subsection 4.3 shall survive termination of this Agreement.

5. **RESTRICTIONS ON USE.**

5.1. Customer may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. Customer shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. Customer shall not make Products available in any form to any person other than Customer's employees and on-site contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use it except as permitted by this Agreement. Customer shall give Mentor Graphics written notice of any unauthorized disclosure or use of the Products as soon as Customer learns or becomes aware of such unauthorized disclosure or use. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive any source code from Software. Log files, data files, rule files and script files generated by or for the Software (collectively "Files"), including without limitation files containing Standard Verification Rule Format ("SVRF") and Tcl Verification Format ("TVF") which are Mentor Graphics' proprietary syntaxes for expressing process rules, constitute or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Customer may use Files containing SVRF or TVF only with Mentor Graphics products. Under no circumstances shall Customer use Software or Files or allow their use for the purpose of developing, enhancing or marketing any product that is in any way competitive with Software, or disclose to any third party the results of, or information pertaining to, any benchmark.

5.2. If any Software or portions thereof are provided in source code form, Customer will use the source code only to correct software errors and enhance or modify the Software for the authorized use. Customer shall not disclose or permit disclosure of source code, in whole or in part, including any of its methods or concepts, to anyone except Customer's employees or contractors, excluding Mentor Graphics competitors, with a need to know. Customer shall not copy or compile source code in any manner except to support this authorized use.

5.3. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense or otherwise transfer the Products, whether by operation of law or otherwise ("Attempted Transfer"), without Mentor Graphics' prior written consent and payment of Mentor Graphics' then-current applicable relocation and/or transfer fees. Any Attempted Transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and/or the licenses granted under this Agreement. The terms

of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer's permitted successors in interest and assigns.

5.4. The provisions of this Section 5 shall survive the termination of this Agreement.

6. **SUPPORT SERVICES.** To the extent Customer purchases support services, Mentor Graphics will provide Customer updates and technical support for the Products, at the Customer site(s) for which support is purchased, in accordance with Mentor Graphics' then current End-User Support Terms located at http://supportnet.mentor.com/about/legal/.

7. **AUTOMATIC CHECK FOR UPDATES; PRIVACY.** Technological measures in Software may communicate with servers of Mentor Graphics or its contractors for the purpose of checking for and notifying the user of updates and to ensure that the Software in use is licensed in compliance with this Agreement. Mentor Graphics will not collect any personally identifiable data in this process and will not disclose any data collected to any third party without the prior written consent of Customer, except to Mentor Graphics' outside attorneys or as may be required by a court of competent jurisdiction.

8. **LIMITED WARRANTY.**

8.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Products will meet Customer's requirements or that operation of Products will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. Customer must notify Mentor Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Software under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes or alternate Software under a transaction involving Software re-mix. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification or improper installation. MENTOR GRAPHICS' ENTIRE LIABILITY AND CUSTOMER'S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY, PROVIDED CUSTOMER HAS OTHERWISE COMPLIED WITH THIS AGREEMENT. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) PRODUCTS PROVIDED AT NO CHARGE; OR (C) BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

8.2. THE WARRANTIES SET FORTH IN THIS SECTION 8 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO PRODUCTS PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

9. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT RECEIVED FROM CUSTOMER FOR THE HARDWARE, SOFTWARE LICENSE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 9 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

10. **HAZARDOUS APPLICATIONS.** CUSTOMER ACKNOWLEDGES IT IS SOLELY RESPONSIBLE FOR TESTING ITS PRODUCTS USED IN APPLICATIONS WHERE THE FAILURE OR INACCURACY OF ITS PRODUCTS MIGHT RESULT IN DEATH OR PERSONAL INJURY ("HAZARDOUS APPLICATIONS"). NEITHER MENTOR GRAPHICS NOR ITS LICENSORS SHALL BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF MENTOR GRAPHICS PRODUCTS IN OR FOR HAZARDOUS APPLICATIONS. THE PROVISIONS OF THIS SECTION 10 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

11. **INDEMNIFICATION.** CUSTOMER AGREES TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH THE USE OF PRODUCTS AS DESCRIBED IN SECTION 10. THE PROVISIONS OF THIS SECTION 11 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

12. **INFRINGEMENT.**

12.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay costs and damages finally awarded against Customer that are attributable to the action. Customer understands and agrees that as conditions to Mentor Graphics' obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance

to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

12.2. If a claim is made under Subsection 12.1 Mentor Graphics may, at its option and expense, (a) replace or modify the Product so that it becomes noninfringing; (b) procure for Customer the right to continue using the Product; or (c) require the return of the Product and refund to Customer any purchase price or license fee paid, less a reasonable allowance for use.

12.3. Mentor Graphics has no liability to Customer if the action is based upon: (a) the combination of Software or hardware with any product not furnished by Mentor Graphics; (b) the modification of the Product other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of the Product as part of an infringing process; (e) a product that Customer makes, uses, or sells; (f) any Beta Code or Product provided at no charge; (g) any software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; or (h) infringement by Customer that is deemed willful. In the case of (h), Customer shall reimburse Mentor Graphics for its reasonable attorney fees and other costs related to the action.

12.4. THIS SECTION 12 IS SUBJECT TO SECTION 9 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS FOR DEFENSE, SETTLEMENT AND DAMAGES, AND CUSTOMER'S SOLE AND EXCLUSIVE REMEDY, WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY PRODUCT PROVIDED UNDER THIS AGREEMENT.

13. **TERMINATION AND EFFECT OF TERMINATION.** If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized term.

13.1. Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement immediately upon written notice if Customer: (a) exceeds the scope of the license or otherwise fails to comply with the licensing or confidentiality provisions of this Agreement, or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. For any other material breach of any provision of this Agreement, Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement upon 30 days written notice if Customer fails to cure the breach within the 30 day notice period. Termination of this Agreement or any license granted hereunder will not affect Customer's obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination.

13.2. Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination, Customer shall ensure that all use of the affected Products ceases, and shall return hardware and either return to Mentor Graphics or destroy Software in Customer's possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form.

14. **EXPORT.** The Products provided hereunder are subject to regulation by local laws and United States government agencies, which prohibit export or diversion of certain products and information about the products to certain countries and certain persons. Customer agrees that it will not export Products in any manner without first obtaining all necessary approval from appropriate local and United States government agencies.

15. **U.S. GOVERNMENT LICENSE RIGHTS.** Software was developed entirely at private expense. All Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to US FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in this Agreement, except for provisions which are contrary to applicable mandatory federal laws.

16. **THIRD PARTY BENEFICIARY.** Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, Microsoft Corporation and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.

17. **REVIEW OF LICENSE USAGE.** Customer will monitor the access to and use of Software. With prior written notice and during Customer's normal business hours, Mentor Graphics may engage an internationally recognized accounting firm to review Customer's software monitoring system and records deemed relevant by the internationally recognized accounting firm to confirm Customer's compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FLEXlm or FLEXnet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics' request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement. The provisions of this Section 17 shall survive the termination of this Agreement.

18. **CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION.** The owners of certain Mentor Graphics intellectual property licensed under this Agreement are located in Ireland and the United States. To promote consistency around the world, disputes shall be resolved as follows: excluding conflict of laws rules, this Agreement shall be governed by and construed under the laws of the State of Oregon, USA, if Customer is located in North or South America, and the laws of Ireland if Customer is located outside of North or South America. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the courts of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply. Notwithstanding the foregoing, all disputes in Asia arising out of or in relation to this Agreement shall be resolved by arbitration in Singapore before a single arbitrator to be appointed by the chairman of the Singapore International

Arbitration Centre ("SIAC") to be conducted in the English language, in accordance with the Arbitration Rules of the SIAC in effect at the time of the dispute, which rules are deemed to be incorporated by reference in this section. This section shall not restrict Mentor Graphics' right to bring an action against Customer in the jurisdiction where Customer's place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

19. **SEVERABILITY.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

20. **MISCELLANEOUS.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements, including but not limited to any purchase order terms and conditions. Some Software may contain code distributed under a third party license agreement that may provide additional rights to Customer. Please see the applicable Software documentation for details. This Agreement may only be modified in writing by authorized representatives of the parties. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

Rev. 100615, Part No. 246066