

Graphical Design Tutorial for HDL Author and HDL Designer

Software Version 2003.1

7 May 2003



**Copyright © Mentor Graphics Corporation 1996-2003.
All rights reserved.**

This document contains information that is proprietary to Mentor Graphics Corporation. The original recipient of this document may duplicate this document in whole or in part for internal business purposes only, provided that this entire notice appears in all copies. In duplicating any part of this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use and distribution of the proprietary information.

This document is for information and instruction purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

RESTRICTED RIGHTS LEGEND 03/97

U.S. Government Restricted Rights. The SOFTWARE and documentation have been developed entirely at private expense and are commercial computer software provided with restricted rights. Use, duplication or disclosure by the U.S. Government or a U.S. Government subcontractor is subject to the restrictions set forth in the license agreement provided with the software pursuant to DFARS 227.7202-3(a) or as set forth in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19, as applicable.

Contractor/manufacturer is:
Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.
Web site: <http://www.hldesigner.com>

This is an unpublished work of Mentor Graphics Corporation.

Table of Contents

About This Manual	viii
Introduction.....	viii
Other Documentation.....	ix
Copying Text From the Acrobat Viewer	xi
Example Designs	xi
Typographic Conventions.....	i-xi
Chapter 1	
VHDL Timer Exercise	1-1
Specification	1-1
Set Library Mapping.....	1-2
Set the Default Language.....	1-4
Create a Block Diagram.....	1-6
Edit the Title Block Template	1-7
Add Blocks.....	1-8
Add Embedded Blocks.....	1-9
Add Ports and Signals	1-10
Add a Bundle and Global Connector	1-12
Save the Block Diagram.....	1-13
Edit the Block and Signal Names.....	1-15
Add an Embedded HDL Text View.....	1-19
Add a Panel and Edit the Title Block.....	1-21
Create a Child State Diagram	1-23
Add States and Transitions.....	1-26
Save the State Diagram	1-27
Edit the States.....	1-28
Edit the Transitions	1-30
Create a Hierarchical State Diagram.....	1-32
Complete the Hierarchical State Diagram.....	1-34
Editing State Machine Properties.....	1-36
Set Generation Properties.....	1-39
Set Checks for HDL Generation	1-41
Generate HDL for the State Machine.....	1-42
Display HDL Files in the Design Explorer	1-43

Table of Contents [continued]

Import the BCDCounter Design Unit.....	1-44
Create a Child Block Diagram.....	1-46
Edit the Generic Mapping.....	1-52
Add ModuleWare Components.....	1-53
Add a User Declaration.....	1-55
Create a Truth Table.....	1-57
Edit the Truth Table.....	1-58
Set Truth Table Properties.....	1-60
Browse the Timer Design.....	1-61
Generate HDL for the Hierarchy.....	1-62
Edit the Timer Symbol.....	1-64
Create a Test Bench.....	1-65
Import the Tester Design Unit.....	1-67
Instantiate the Imported Tester.....	1-68
Generate HDL for the Test Bench.....	1-70
Browse the Completed Design.....	1-72
Setup the Downstream Tools.....	1-73
Run the ModelSim Flow.....	1-74
Setup the Simulator Windows.....	1-76
Add Simulation Probes.....	1-77
Enable Animation.....	1-78
Simulate the Design.....	1-80
Review the Animation.....	1-82
Debugging From ModelSim.....	1-83
Run the LeonardoSpectrum Flow.....	1-85
Using the Example VHDL Design.....	1-89
Chapter 2	
Verilog Timer Exercise.....	2-1
Specification.....	2-1
Set Library Mapping.....	2-2
Set the Default Language.....	2-4
Create a Block Diagram.....	2-6
Edit the Title Block Template.....	2-7
Add Blocks.....	2-8

Table of Contents [continued]

Add Embedded Blocks.....	2-9
Add Ports and Signals.....	2-10
Add a Bundle and Global Connector.....	2-12
Save the Block Diagram.....	2-13
Edit Block and Signal Names.....	2-15
Add an Embedded HDL Text View.....	2-19
Add a Panel and Edit the Title Block.....	2-21
Create a Child State Diagram.....	2-23
Add States and Transitions.....	2-26
Save the State Diagram.....	2-27
Edit the States.....	2-28
Edit the Transitions.....	2-30
Create a Hierarchical State Diagram.....	2-32
Complete the Hierarchical State Diagram.....	2-34
Editing State Machine Properties.....	2-36
Set Generation Properties.....	2-39
Set Checks for HDL Generation.....	2-41
Generate HDL for the State Machine.....	2-42
Display HDL Files in the Design Explorer.....	2-43
Import the BCDCounter Design Unit.....	2-44
Create a Child Block Diagram.....	2-46
Edit the Parameter Mapping.....	2-52
Add ModuleWare Components.....	2-53
Add a User Declaration.....	2-55
Create a Truth Table.....	2-57
Edit the Truth Table.....	2-58
Set Truth Table Properties.....	2-59
Add a Module Declaration.....	2-61
Browse the Timer Design.....	2-62
Generate HDL for the Hierarchy.....	2-63
Edit the Timer Symbol.....	2-65
Create a Test Bench.....	2-66
Import the Tester Design Unit.....	2-68
Instantiate the Imported Tester.....	2-69
Generate HDL for the Test Bench.....	2-71

Table of Contents [continued]

Browse the Completed Design	2-73
Setup the Downstream Tools	2-74
Run the ModelSim Flow	2-75
Setup the Simulator Windows	2-77
Add Simulation Probes	2-78
Enable Animation	2-79
Simulate the Design	2-81
Review the Animation	2-83
Debugging From ModelSim	2-84
Run the LeonardoSpectrum Flow	2-86
Using the Example Verilog Design	2-90
Chapter 3	
Creating a VHDL Flow Chart	3-1
Introduction	3-1
Create the Tester Flow Chart	3-2
Set Flow Chart Properties	3-4
Add a Start Point and Action Box	3-7
Add a Loop and an Associated Comment	3-8
Add an Action Box	3-11
Add a Hierarchical Action Box	3-12
Add a Decision Box	3-14
Add Wait Boxes	3-16
Copy the Decision Tree	3-18
Completing the Flow Chart	3-19
Using HDL2Graphics	3-21
Chapter 4	
Creating a Verilog Flow Chart	4-1
Introduction	4-1
Create the Tester Flow Chart	4-2
Set Flow Chart Properties	4-4
Add a Start Point and Action Box	4-6
Add a Loop and an Associated Comment	4-8

Table of Contents [continued]

Add an Action Box.....	4-11
Add a Hierarchical Action Box.....	4-12
Add a Decision Box	4-14
Add a Wait Box.....	4-16
Copy the Decision Tree.....	4-18
Completing the Flow Chart.....	4-19
Using HDL2Graphics	4-21

Trademark Information

End-User License Agreement

About This Manual

Introduction

This manual provides a self-paced tutorial with step-by-step procedures for creating a simple timer design and test bench using VHDL or Verilog graphical views.

You are advised to perform the separate the *Design Management Tutorial* before attempting this tutorial to learn about creating projects and managing your design data.

You can learn about using tabular Interface-Based Design views by performing the separate *Interface-Based Design Tutorial*.

Procedures for completing the tutorial using VHDL are given in [Chapter 1](#) and procedures for using Verilog in [Chapter 2](#). If you want to complete both versions of the tutorial, you should create a separate library for each version of the design.

The tutorial covers the basic procedures required to fully define and verify a design using graphical views. The full tutorial can be completed by users of the HDL Author or HDL Designer tools using block diagram, symbol, state diagram, truth table and flow chart views.

The completed design can be compiled and simulated if ModelSim is available and synthesized if the LeonardoSpectrum tools are available.

The main tutorial includes a test bench which uses a flow chart view. The flow chart can be imported from example HDL code or can optionally be created by following the procedures in [Chapter 3](#) and [Chapter 4](#).

Although the procedures do not describe the use of other tools, the HDL Designer Series includes support for many alternative downstream tool interfaces and it should be possible to use the generated HDL with any of these interfaces.

However, you must consider any limitations of your external tool. In particular, some VHDL tools may not support the standard IEEE packages used in the tutorial and you should substitute an appropriate alternative package.

The tutorial assumes that users have some knowledge of the issues for digital hardware design and experience of the VHDL or Verilog language.

It is possible to complete the tutorial without this knowledge by carefully copying the language syntax given in the procedures. However, a separate VHDL or Verilog training course is recommended in order to fully appreciate how the power of HDL design can be exploited using graphical design methods.



The illustrated examples have been laid out for maximum readability in the Acrobat viewer or on the printed page. When you are creating your own design, it is advisable to allow extra space between diagram objects so that you can easily route signals between them.

All user commands in the tutorial procedures are referenced using menu path (shown in bold text) or toolbar button. However, many commands can also be accessed using the shortcut bar or keyboard shortcuts. Refer to **Shortcut Keys** in the **Quick Reference Index** which can be accessed from the **Help** menu for lists of the available shortcut keys.

The tutorial is divided into sections containing numbered procedures which must be performed for the each section of the tutorial to be completed successfully. If you are working from a printed copy of the tutorial, you are recommended to tick off each numbered procedure as you complete it to ensure no steps are omitted.

Unnumbered paragraphs provide additional information or give advice on other ways to perform an operation. The procedure numbers are continuous through each major section and its subsections but restart for each major section, for example when introducing a new editor.

Other Documentation

The HDL Designer Series documentation (including the release notes and this user manual) can be accessed from the *HDL Designer Series Bookcase* which is available from the **Help** menu in each application window.

Self-paced tutorials can be accessed from the *HDL Designer Series Tutorials* bookcase which can be accessed from the main bookcase or directly from the **Help** menu.

All documents are provided in Adobe Acrobat portable document format (PDF). The Acrobat reader can be installed from the HDL Designer Series CD-ROM if it is not already available.

Refer to the *Release Notes* for information about new features or upgrading from a previous product.

Refer to the *Start Here Guide for the HDL Designer Series* for information about the HDL Designer Series tools, supported platforms and configurations.


Refer to the *HDL Designer Series User Manual* for information about the design manager and procedures common to all HDL Designer Series tools.

Refer to the *HDL Designer Series Graphical Editors User Manual* for information about the graphical editors.

Refer to the *DesignPad Text Editor User Guide* for information about the built in HDL text editor.





Refer to the *Debug Detective User Guide* for information about Debug Detective.



Refer to the *ModuleWare Reference Guide* for information about the ModuleWare library.

A number of quick reference help pages can be accessed by choosing the **Quick Reference Index** from the **Help** menu. Further quick reference pages can be accessed using the  buttons provided on many of the application dialog boxes.

Information about the state diagram, flow chart and truth table editors is provided as help topics which can be accessed by choosing **Help Topics** from the **Help** menu in these editors.

Copying Text From the Acrobat Viewer

You can copy text from this document by choosing the  (**Text Select**) tool button or  shortcut key in the Acrobat viewer and choosing **Copy** from the Acrobat **Edit** menu (or using the + shortcut).

The text can be pasted into a text editor (or application dialog box) using the + shortcut or the **Paste** menu option if one is provided in the destination window. In the graphic editors, you can use the **Paste Special** option to explicitly paste text from the system clipboard.



If you copy HDL text from a tutorial help page, check that punctuation characters are copied correctly. In particular, line feed characters may not be translated on UNIX systems and may need to be re-entered.

Example Designs

Completed examples of the VHDL and Verilog tutorial designs are provided in the HDL Designer Series installation. These can be browsed for reference or can be compiled, simulated and animated directly if you modify their downstream library mapping to use a writable downstream directory.

Refer to the sections [“Using the Example VHDL Design” on page 1-89](#) or [“Using the Example Verilog Design” on page 2-90](#) for more information.

Typographic Conventions

The following conventions have been used in this manual:

Style	Usage
Bold	Menu options and command line switches.
<i>Italic</i>	Pathnames (and object names derived from file names). Also used for manual titles (for example, <i>ModuleWare Reference Guide</i>).

Style	Usage
<code>Courier font</code>	Monospaced Courier font is used for code examples.
Links	Cross references within the text are shown in blue.



When pathnames (or window titles derived from pathnames) are shown in this tutorial, the PC convention (\) is used.

Chapter 1

VHDL Timer Exercise

This exercise creates a simple timer using block diagrams and a control block described as a hierarchical state machine. A simple truth table is used to decode four-bit binary codes from the ten-bit input bus. The design is completed using a re-usable component described by a HDL text view.

A test bench is created using a flow chart which can be used as a test harness to simulate the generated VHDL for the timer design. The simulation results can be displayed as animation on the flow chart and state machine to assist in debugging the design. The verified timer design is then synthesized.

The instructions assume that a *ModelSim* simulator and the LeonardoSpectrum synthesis tools are available. However, the VHDL generated from the diagrams can also be used by other compatible downstream tools that are available on your system.

Specification

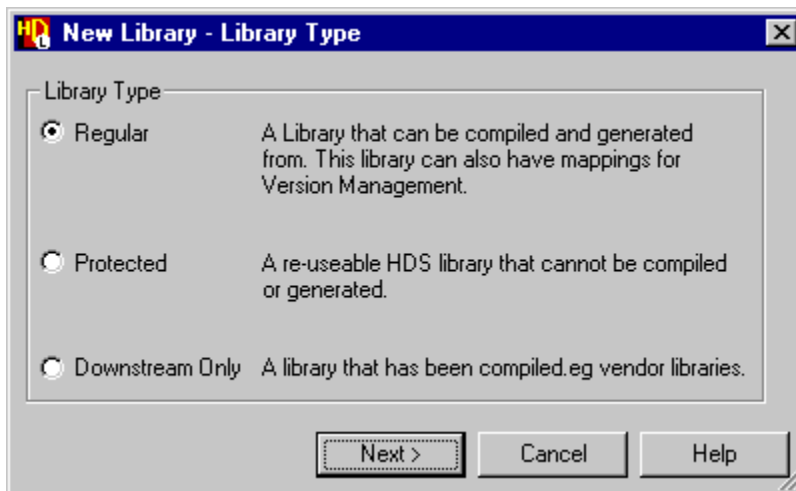
The timer outputs time data on two four-bit buses representing low and high values. There is also a logic output signal which triggers an audible alarm. The data input is provided on a ten-bit bus and control is provided by start, stop, reset and clock signals. These signals are summarized in the following table:


Inputs	Outputs
start (logic signal)	high (4-bit bus)
stop (logic signal)	low (4-bit bus)
reset (logic signal)	alarm (logic signal)
clk (logic signal)	
d (10-bit bus)	

Set Library Mapping

For this tutorial, you can use an existing project such as the project you created for the Design Management tutorial but should create a new library mapping for your design data. The library mapping defines the logical location of the directories containing your design data. The source graphical objects, HDL and downstream data can be stored at any writable locations on your available file system but are typically saved below a common root directory.

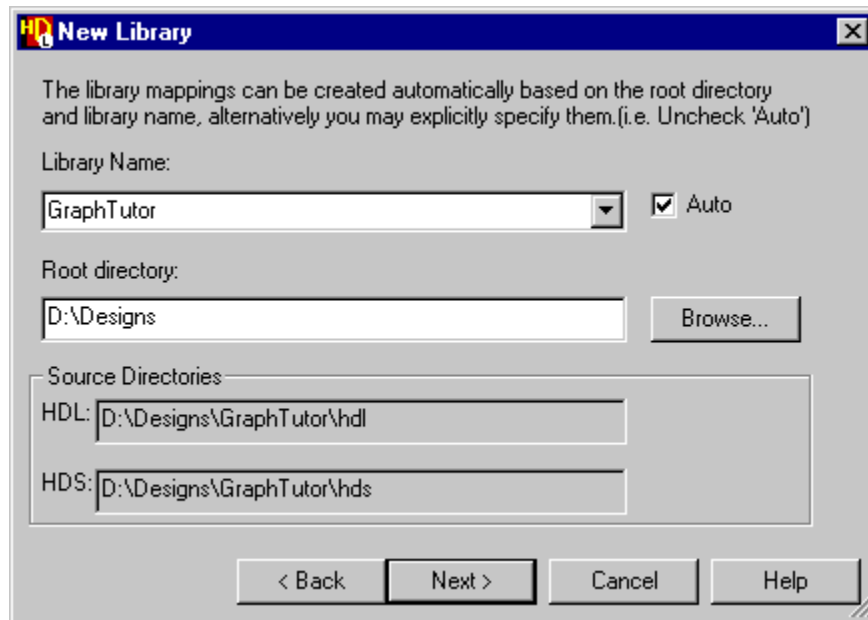
1. To set library mapping, click the  **New Library** icon in the Project group on the shortcut bar or choose **Library** from the **New** cascade of the **File** menu in the design manager window to display the New Library wizard:



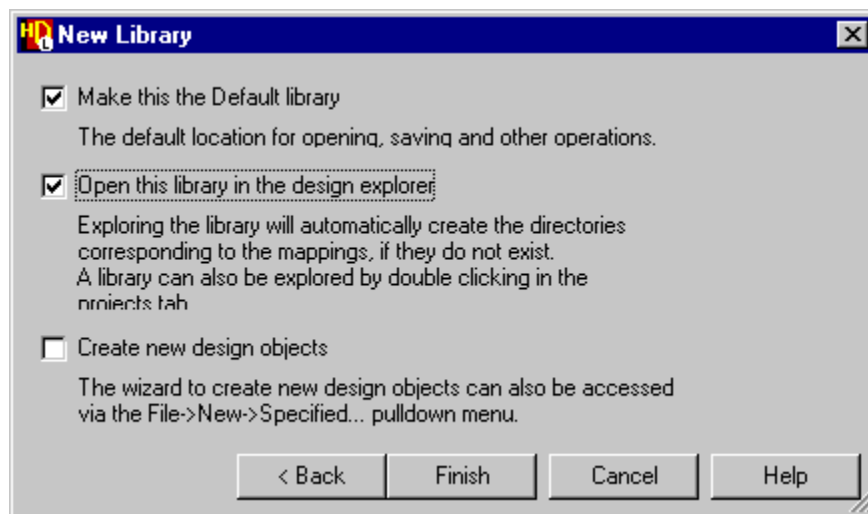
2. Choose the **Regular** library type and click the  button to display the library name page.
3. Enter a logical library name (for example: *GraphTutor*) and check that the **Auto** option is set in the dialog box.
4. Specify or browse for the pathname for the root directory that will contain your library data (for example, *D:\Designs* or *\$HOME/designs*).

Library names and pathnames can be entered using upper, lower or mixed case but note that UNIX systems are case sensitive and the case used for pathnames should match the file structure. (On a PC, library names are case sensitive but pathnames are case insensitive.)

Notice that the source HDL directory is named `<root_directory>\GraphTutor\hdl` and the source graphics directory is named `<root_directory>\GraphTutor\hds`.

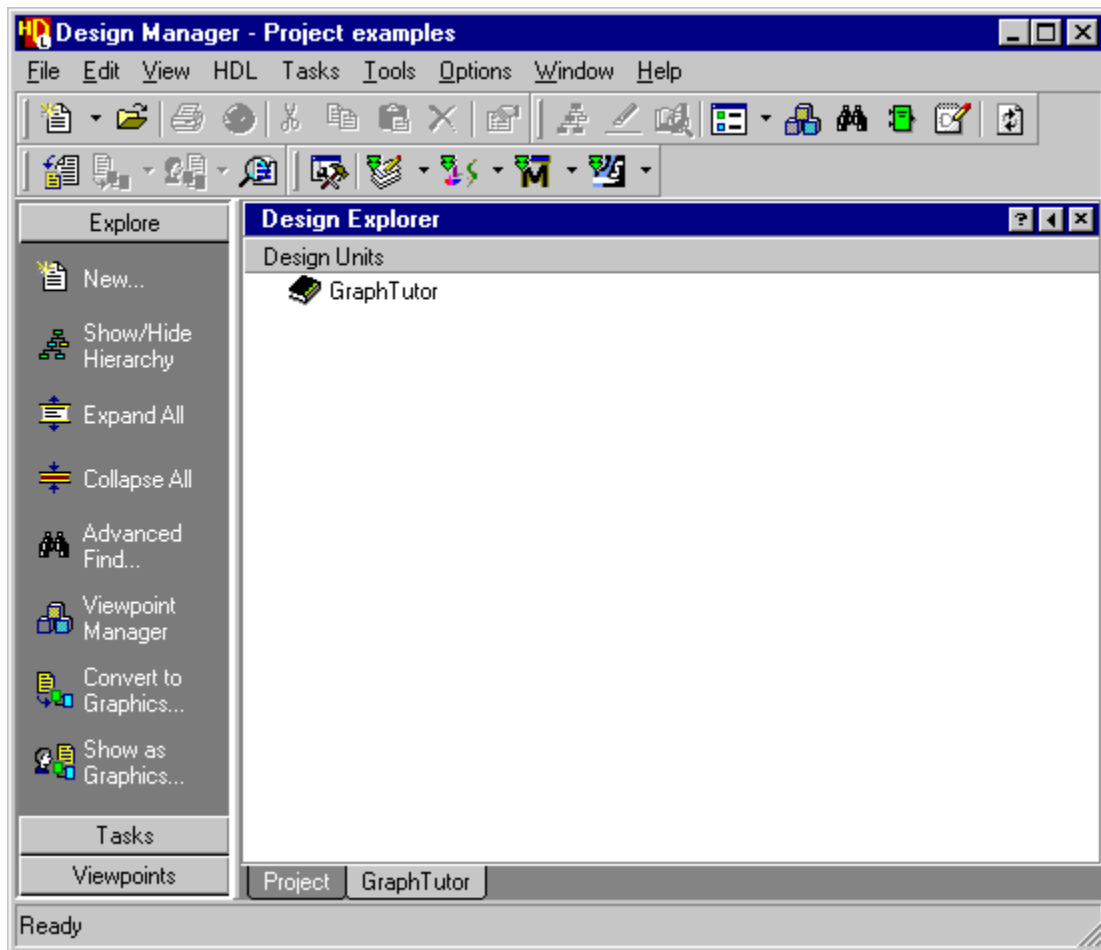


5. Click the **Next >** button to display the last page of the wizard.



6. Set the **Make this the Default library** and **Open this library in the design explorer** options.
7. Click the **Finish** button to close the wizard.

The source design data directories you specified in the wizard are created and the library (shown as a closed “book”) is opened in a new design explorer window:

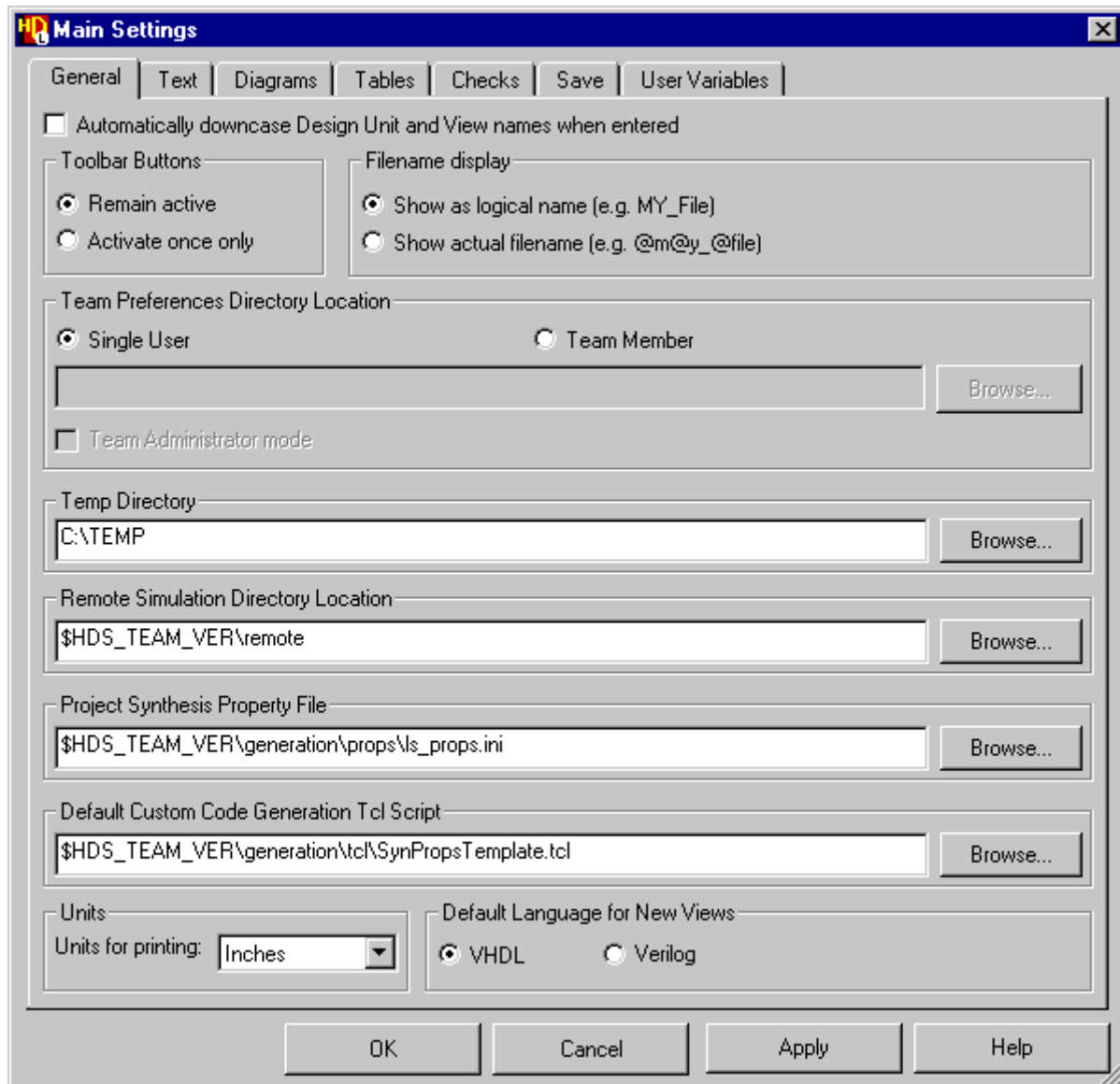


The mapping for the location of compiled data will be defined automatically when the design is compiled.

Set the Default Language

A set of default preferences are loaded when you invoke a HDL Designer Series tool for the first time. There are separate tabbed dialog boxes for the main settings, VHDL and Verilog options, HDL2Graphics import options, version management, animation settings and master preferences for each type of graphical diagram. The preference dialog boxes can be accessed from the **Options** menu.



1. Choose **Main** from the **Options** menu to display the Main Settings dialog box.

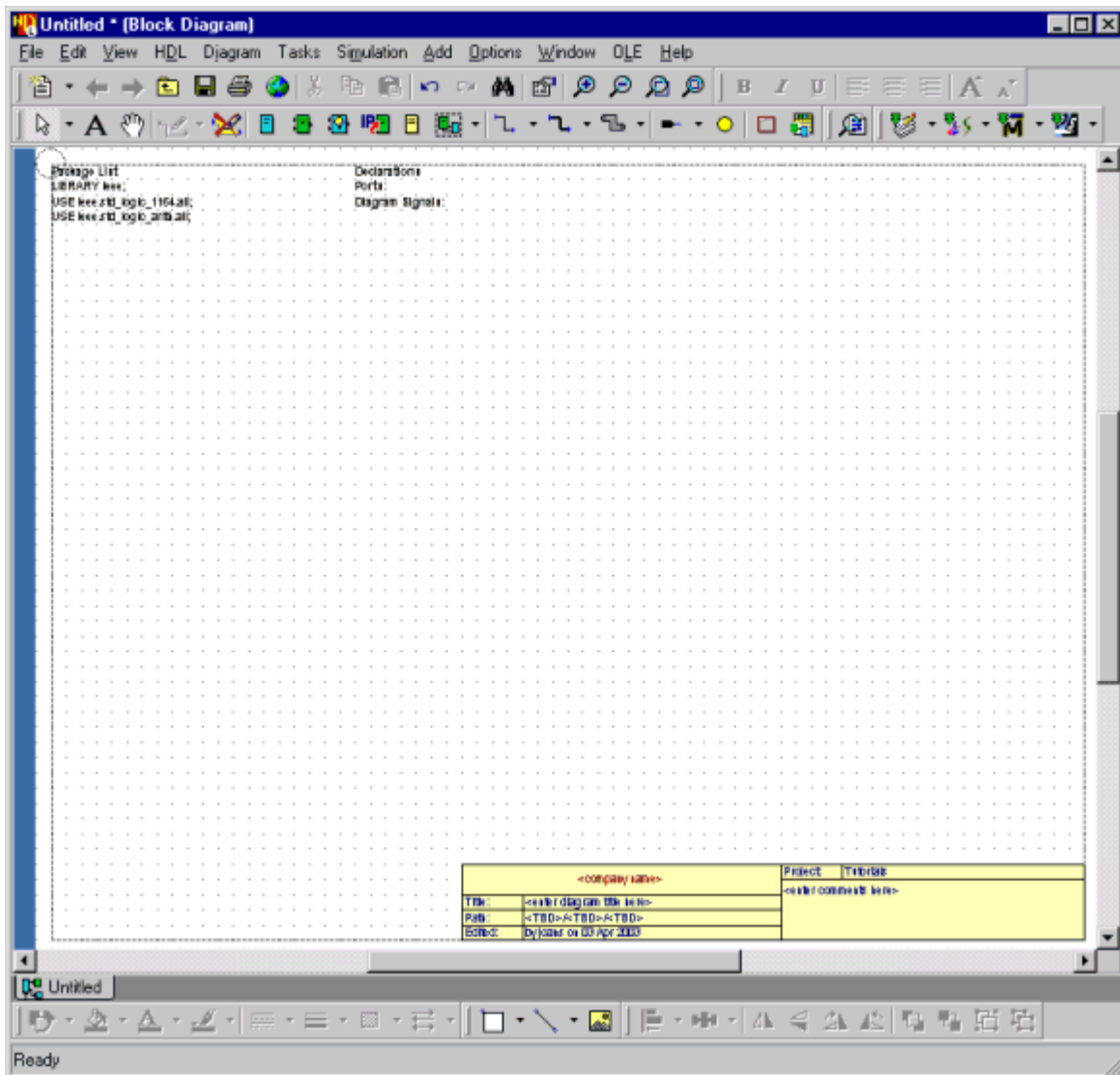


2. Select the **General** tab and ensure that **VHDL** is set as the default language to be used for new graphic editor views. Use the **OK** button to confirm your choice.

All other preferences can be left with their default values for this tutorial.

Create a Block Diagram

1. Use the  button in the design manager window and select **Block Diagram** from the **Graphical View** cascade of the dropdown menu to create a new untitled block diagram.
2. Use the  button to view the entire diagram:.



Notice the five toolbars at the top and three toolbars at the bottom of the diagram. The toolbar buttons provide quick access to many of the most frequently used editing and formatting commands.

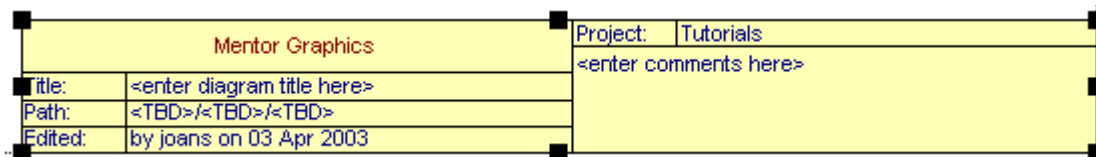
The block diagram is a blank sheet except for a background grid, a package list (with the standard IEEE libraries *std_logic_1164* and *std_logic_arith*) and empty text fields with labels for *Declarations*, *Ports* and *Diagram Signals*.

The diagram also shows page boundaries for the default printer and a copy of the default title block.

Edit the Title Block Template

A title block is automatically added to all new diagrams if the **Add Title Blocks in new diagrams** option is set in your diagram preferences. The default title block is a template with default locations for your company name, diagram title and comments. The template incorporates internal variables which automatically enter the current project name, your login name and the current date. Internal variables are also used to enter the logical pathname for the design. This path is initially shown as <TBD>/<TBD>/<TBD> but the internal variables are converted to show the library, design unit and view name when you save the diagram.

- Click twice on <company name> in the default title block to display a popup edit box and replace the default text by the name of your company.
- Select the title block by clicking with the mouse so that the selection handles are displayed and choose **Save Title Block** from the **File** menu.



- A dialog box is displayed with a warning that saving the title block cannot be undone. Click the button to proceed.

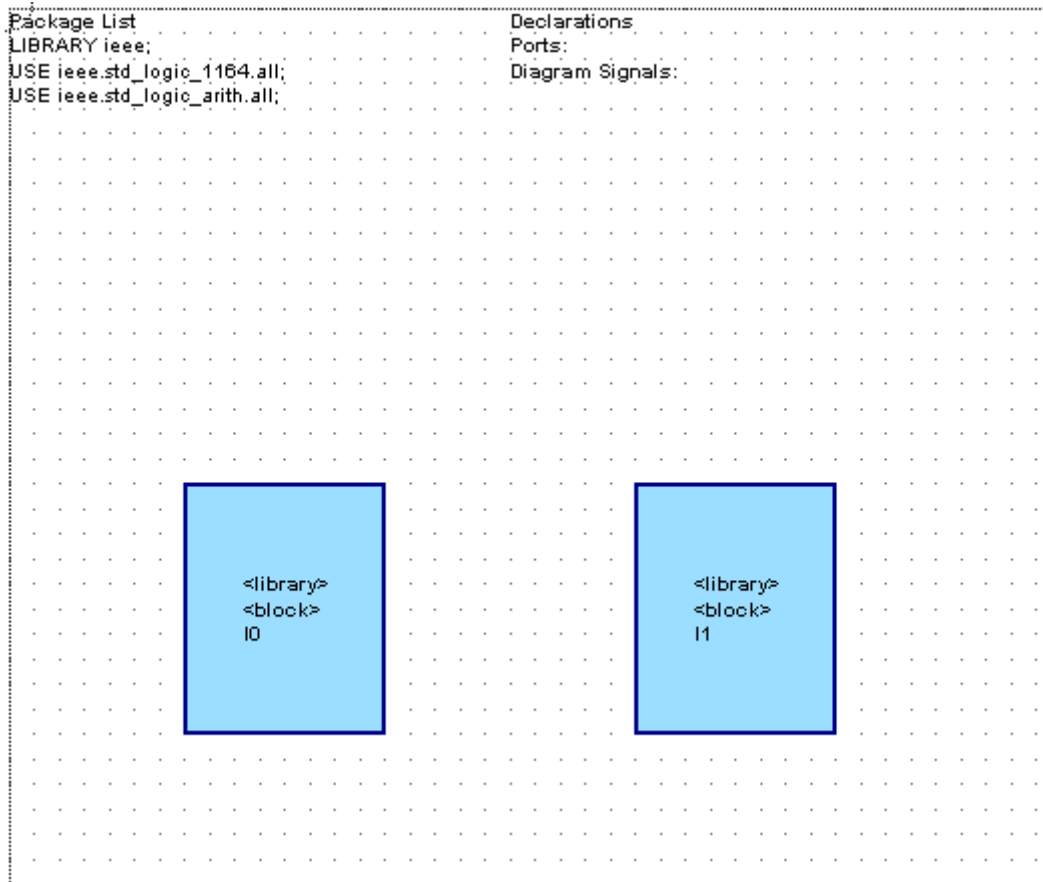


The title block is saved as a team resource at the location specified in your preferences and will be used as the default template in new diagrams. If shared team preferences have been set up on your system, there may already be a read-only team template stored at this location.

Refer to the *HDL Designer Series User Manual* for information about how you can add and modify title blocks.


Add Blocks




6. Use the  button (or choose **Block** from the **Add** menu) to add two blocks on the diagram as shown below:




Notice that the blocks are added with the default library *<library>*, the default name *<block>* and unique instance names (*I0* and *I1*).

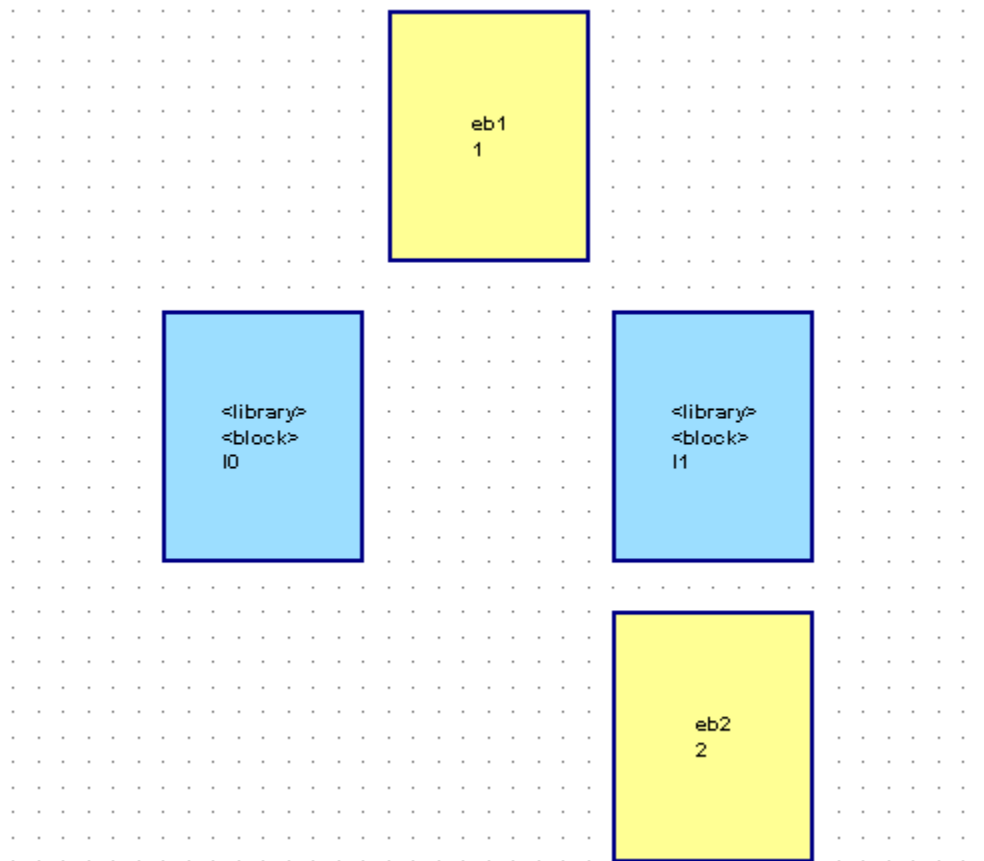


The Add Block command normally auto-repeats until you select another command or terminate the repeating command by using the right mouse button or  key. However, you can change the behavior of the toolbar buttons by setting the Activate once only preference in the General tab of the Main Settings dialog box.

You can also use the  key with any toolbar button to toggle the repeat mode. For example, when Remain active is set,  +  adds a single block on a block diagram.

Add Embedded Blocks

- Use the  button to add two embedded blocks on your block diagram as shown in the picture below.








Notice that the embedded blocks are added with unique default names (*eb1* and *eb2*) and numbers (*1* and *2*).


The view describing a block must be saved as a uniquely named design unit in a library directory. However, the view describing an embedded block is saved as part of the parent block diagram and does not impose hierarchy when HDL is generated for your design. The name of an embedded block must be unique on the diagram and is used as a label in the generated HDL.




The blocks (*I0* and *I1*) will be used to define a child state machine and block diagram view. The embedded blocks (*eb1* and *eb2*) will be defined by concurrent assignment statements on the top level block diagram.


Add Ports and Signals

You can use the following buttons to add signal and bus nets on a block diagram:


- | | |
|---|--------------------------|
|  | Add a signal |
|  | Add a signal with a port |
|  | Add a bus |
|  | Add a bus with a port |
|  | Add a bus with a ripper |

Signals or buses can be added between any existing connectable items on the diagram or left unconnected by double-clicking to terminate the net with a dangling net connector. However, you can use the  pulldown on the buttons to change the default setting and terminate with a default port or ripper. Notice that the toolbar button changes to show the current setting.


When the  or  button is selected, a port is automatically added at an unconnected source or destination end point. When the  button is selected, a ripper is used if the end point is over an existing bus or bundle.

- Choose **Signal with Port** and use the  button to connect three signals originating from the block on the left (instance *I0* in the picture) to the block on the right (instance *I1*) and one signal returning from *I1* to *I0*. The signals are added with unique names (*sig0*, *sig1*, *sig2* and *sig3*) and the default type *std_logic*. Notice how the full declarations for these diagram signals are automatically added to the list of Declarations on the diagram.

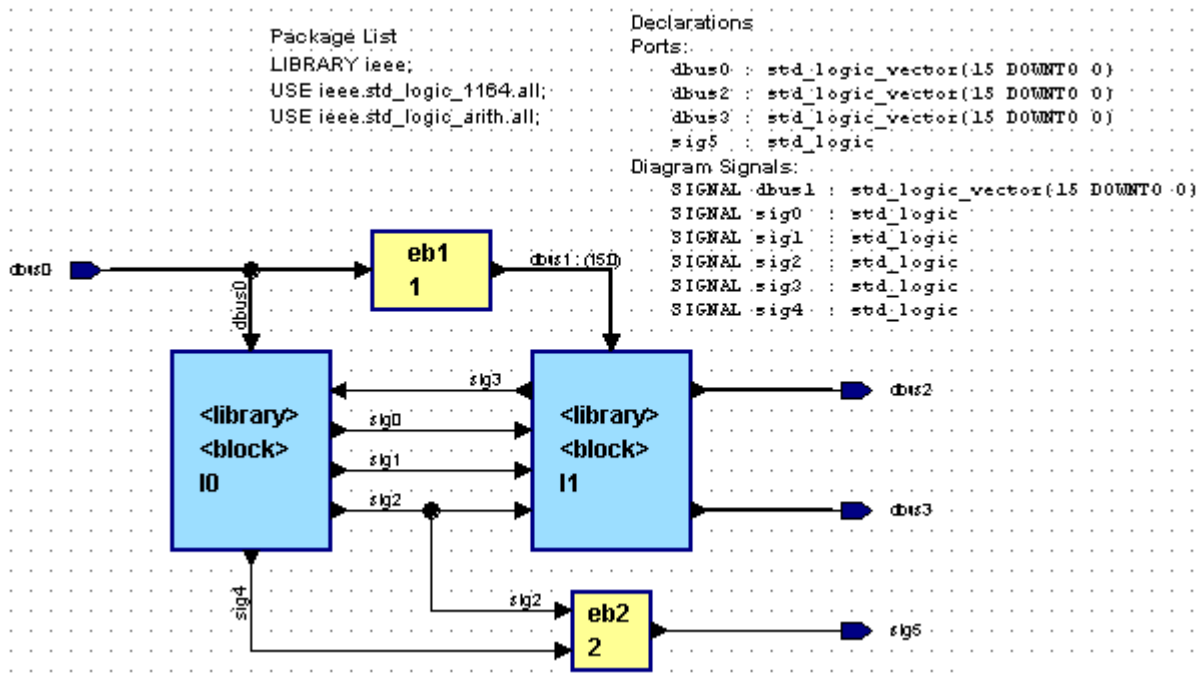


Allow one or more grid lines between each port or signal. You can resize objects by selecting a block or embedded block and dragging one of its resize handles. If necessary, you can drag text elements such as the signal name using the  mouse button.




- Add a signal from *I0* to the embedded block *eb2* and another signal from a point on *sig2* terminating on the embedded block.

10. Add a signal from the embedded block *eb2* terminating in space on the right side of your diagram. Notice that an output port is added when you double-click at the end of the last signal and its declaration is added to the list of ports on the diagram.
11. Choose **Bus with Port** and use the  button to add a bus from a source on the left side of your diagram with its destination on the upper embedded block *eb1*. A default input port is automatically created at the beginning of the bus.
12. Add another bus starting from this bus and terminating on instance *I0*. Notice how both bus segments have the same default name *dbus0*. The full declaration showing the default bus type and bounds *std_logic_vector(15 DOWNTO 0)* is added to the list of ports.
13. Add a bus (*dbus1*) from *eb1* to *I1*. This internal diagram signal is shown with the default bounds (*15:0*) shown (in abbreviated format) on the net.
14. Then add two buses (*dbus2* and *dbus3*) from *I1* terminated with default output ports on the right of the diagram.





Your diagram should now look similar to the picture below:





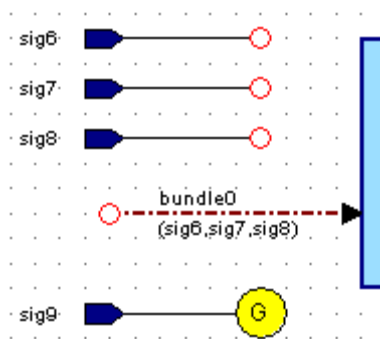
Add a Bundle and Global Connector




15. Use the  button to add three signals on the left side of your diagram. Notice that a default input port is created at the source of each signal but a dangling net connector is drawn when you double-click at the end of each signal.
16. Select the three signals (by dragging a select rectangle with the  mouse button held down) and use the  button to connect a bundle containing these signals to block instance *I0* as shown in the picture below.

Notice that the bundle has the default name *bundle0* and the three selected signals are automatically included in the bundle with their names listed under the bundle name.

 You can use the  pulldown on the  button to select a  button which allows you to rip one or more signals and buses from an existing bundle.


17. Use the  button to add a global connector on your diagram below the bundle.
18. Use the  button to add a signal between the global connector and a default input port. (This will be a clock signal which is implicitly connected to every block on the diagram.)

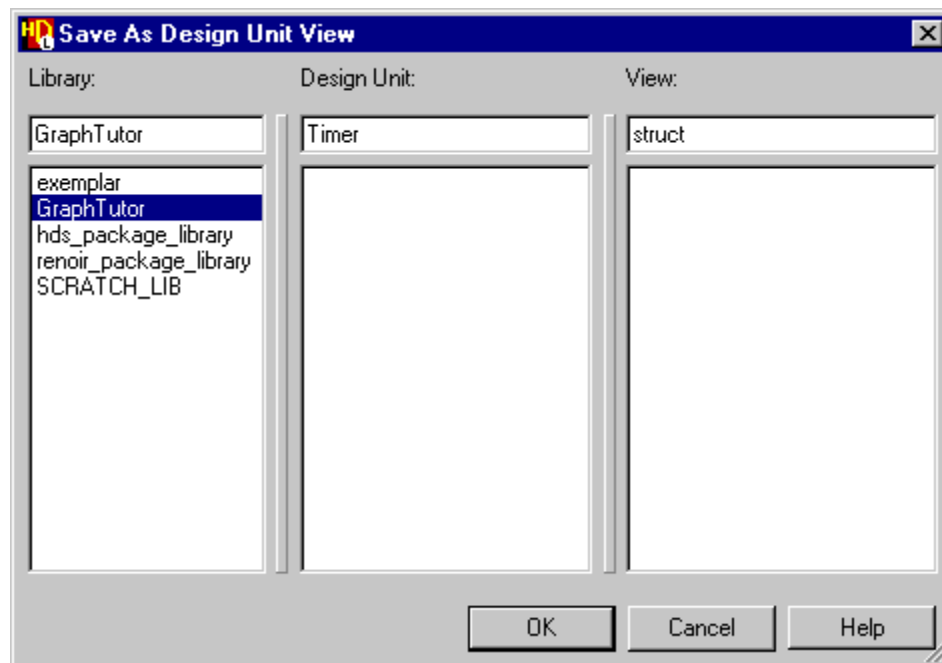


 If you make a mistake when editing a diagram, you can use the  button to undo your last edit and the  button to redo an undo operation. You can also use commands from the **Align** cascade of the **Edit** menu to re-align and distribute objects on the diagram.

Save the Block Diagram

Notice the asterisk (*) character in the header of the block diagram editor window. This indicates that the diagram has been edited since it was last saved.

19. Use the  button to save the block diagram. The Save As dialog box is displayed which allows you to choose from the currently mapped libraries and specify the design unit and design unit view names.
20. Choose the *GraphTutor* library and enter design unit *Timer*. The dialog box should look similar to the example below:



The view name and file extension when you save graphical diagrams defaults as follows:

struct.bd	block diagram
struct.ibd	interface-based design view
fsm.sm	state diagram
flow.fc	flow chart
tbl.tt	truth table
symbol.sb	symbol



If you omit the two-character extension it is automatically added to identify the type of diagram you are saving. The default leaf names can be changed by setting preferences. However, you should not change the extension (*.bd*, *.ibd*, *.sm*, *.fc*, *.tt* or *.sb*) or the design data file will not be recognized and cannot be reopened.

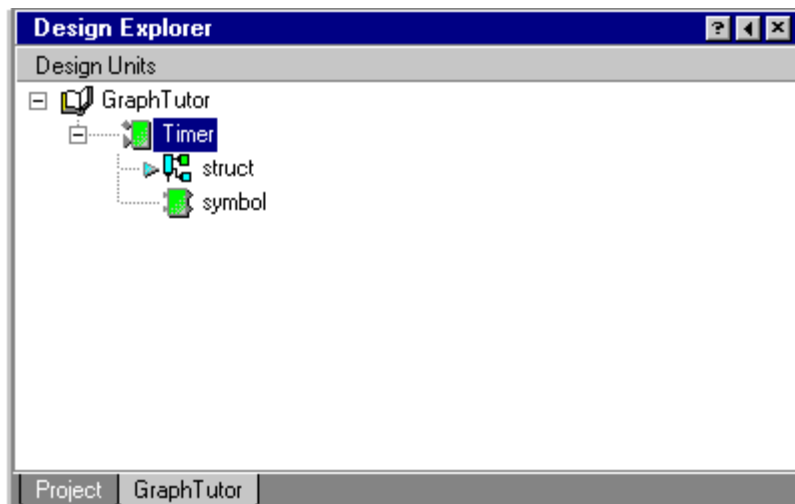
21. When you click the button, your diagram is saved and the window title bar is updated to show the pathname *GraphTutor/Timer/struct*.

This path is also added in the title block replacing the <TBD> used when the diagram was created. Notice that the asterisk (*) character has been cleared in the block diagram header and the library name used on the blocks in your diagram has been updated to *GraphTutor*.



If the design manager window is obscured, you can pop it to the front by selecting **Design Manager** from the list of open windows in the block diagram **Windows** menu.

22. Click on the  icon for the *GraphTutor* library in the design explorer and notice that the view is expanded to display the *Timer* design unit.
23. Click on the  icon for the *Timer* design unit to reveal that it contains a symbol and block diagram view.



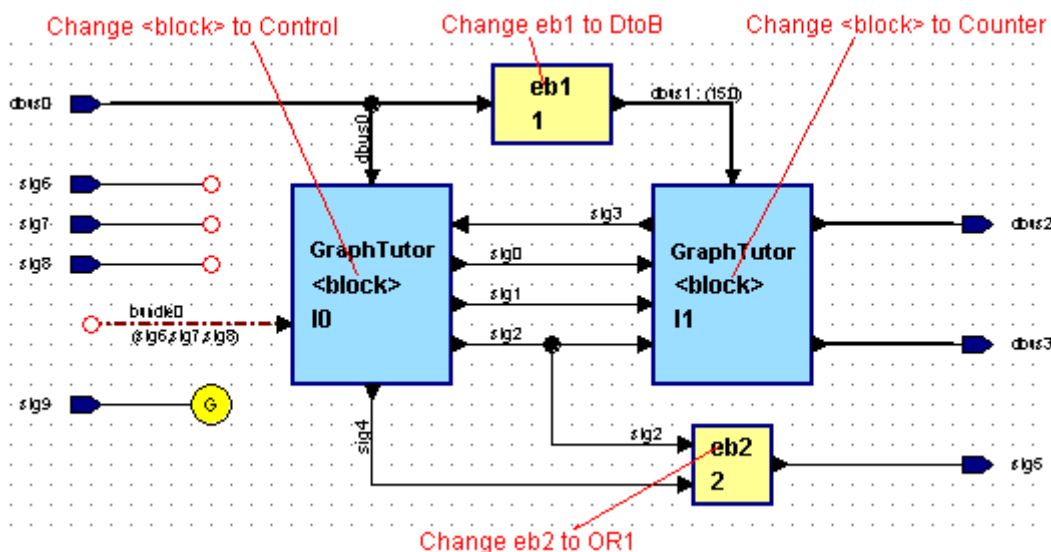
Edit the Block and Signal Names

You now have a completed top-level block diagram for the *Timer* design. However, the blocks and signals have default names.

- Click on the text `<block>` in the lower block on the left (instance *I0* in the picture) and notice the small handles which indicate that the text object is selected. Click again and notice that the text is now highlighted and can be directly overwritten.


If you click again, the cursor changes to an I-beam which allows you to move the cursor in the text and edit individual characters. Enter the new name *Control* and click outside the text to complete the edit.

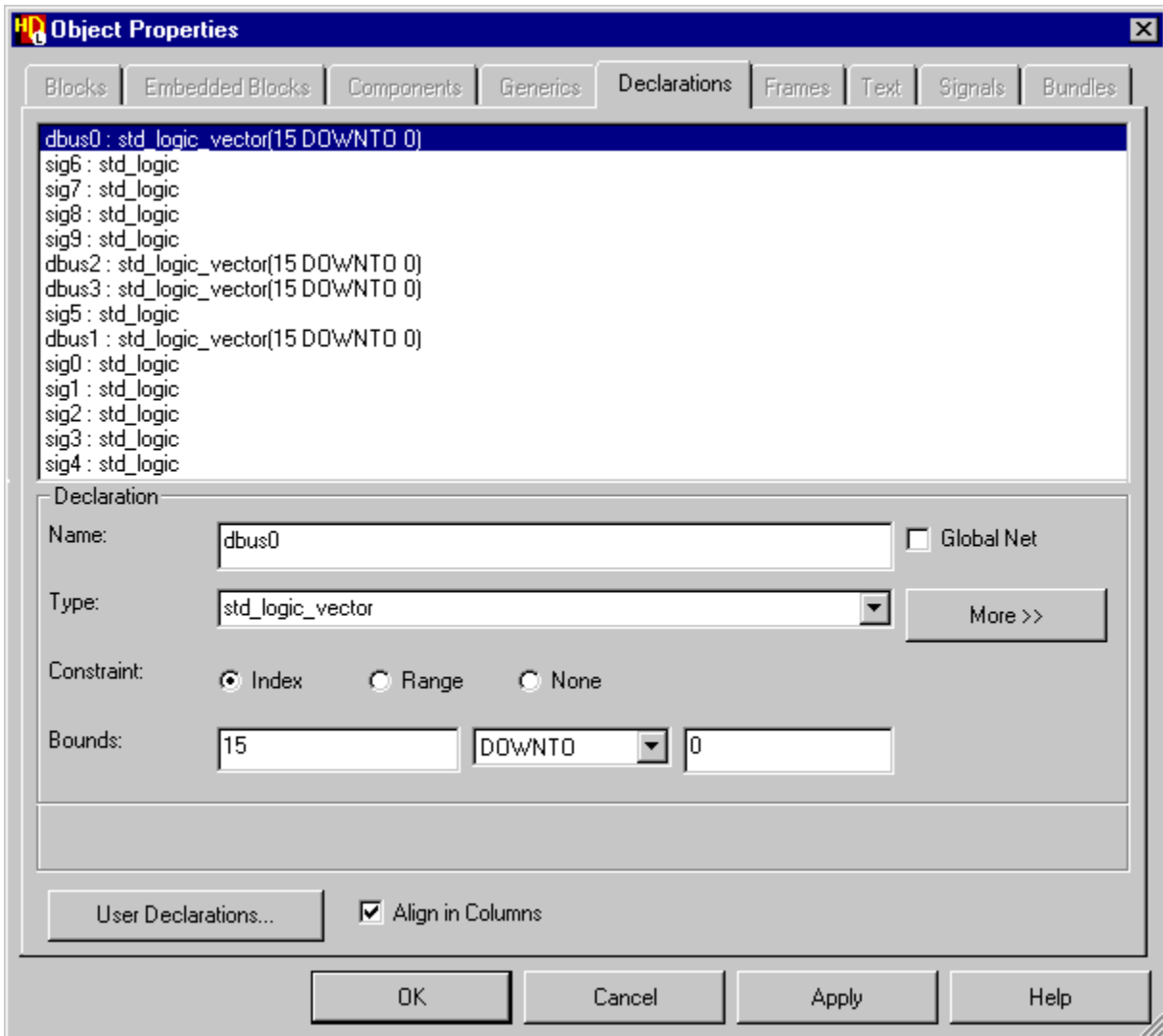
- Repeat this procedure to change the name of block instance *I1* to *Counter*, embedded block *eb1* to *DtoB* and embedded block *eb2* to *OR1*.



Direct text editing can also be used to edit the signal and bus names. Alternatively, you can use a dialog box which allows you to edit the properties for a selected object. By default, edits to signal and bus nets are applied only to the connected nets but you can choose to apply the changes to the entire diagram or to propagate changes to all occurrences of the net in the hierarchy of the design.



- Choose **Entire Net in diagram** from the **Scope for Changes** cascade of the **Signals** cascade in the **Diagram** menu.

27. Double-click on the existing declarations, use the  button or choose **Object Properties** from the **Edit** menu to display the Block Diagram Object Properties dialog box and choose the **Declarations** tab.



Notice that the port declarations are listed at the top of the dialog box and the other internal diagram signals at the bottom. Input ports are listed before the output ports, otherwise the declarations are listed in alphanumeric order.

You can choose one or more existing declarations in the dialog box and enter new values for any of the declaration fields. For example, use **Ctrl** + **Left** mouse button to choose *dbus1*, *dbus2* and *dbus3*, then enter a new *index* constraint with bounds *3 DOWNTO 0* to update all three buses while all other fields remain AS_IS.

The changes are applied to the diagram when you click the  or  button. Notice that all occurrences on the diagram are updated including the declarations list, signals, buses and bundle contents and that the lists of port and signal declarations are sorted alphanumerically when the changes are applied to the diagram.

28. Use the dialog box to update the port and signal declarations as shown in the following tables.

Ports:

Old Name	New Name	Type	Constraint	Bounds
dbus0	d	std_logic_vector	index	9 DOWNTO 0
sig6	start	std_logic	none	none
sig7	stop	std_logic	none	none
sig8	reset	std_logic	none	none
sig9	clk	std_logic	none	none
dbus2	low	std_logic_vector	index	3 DOWNTO 0
dbus3	high	std_logic_vector	index	3 DOWNTO 0
sig5	alarm	std_logic	none	none

Diagram Signals:



Old Name	New Name	Type	Constraint	Bounds
dbus1	dat_in	std_logic_vector	index	3 DOWNTO 0
sig0	clear	std_logic	none	none
sig1	load	std_logic	none	none
sig2	hold	std_logic	none	none
sig3	zero	std_logic	none	none
sig4	beep	std_logic	none	none



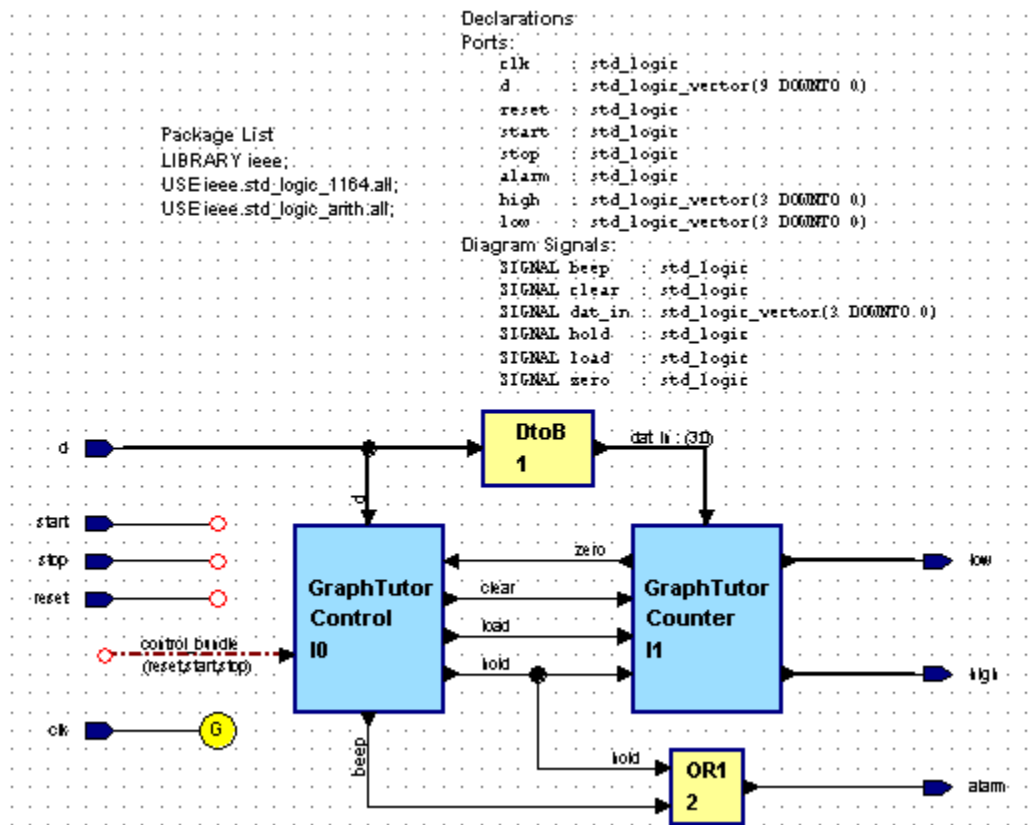
All occurrences of each signal name (including the bundle contents) should be automatically updated on the diagram when you confirm the dialog box. If any nets are not updated, check that you have set the scope for changes to **Entire Net in Diagram** as described on [page 1-15](#).



- Select the bundle name and use direct text editing or the **Bundles** tab of the Object Properties dialog box to change the bundle name to *control_bundle*.



If you have difficulty selecting text objects, you can change the selection mode by using the  pull-down on the  button to select text or object shapes only.

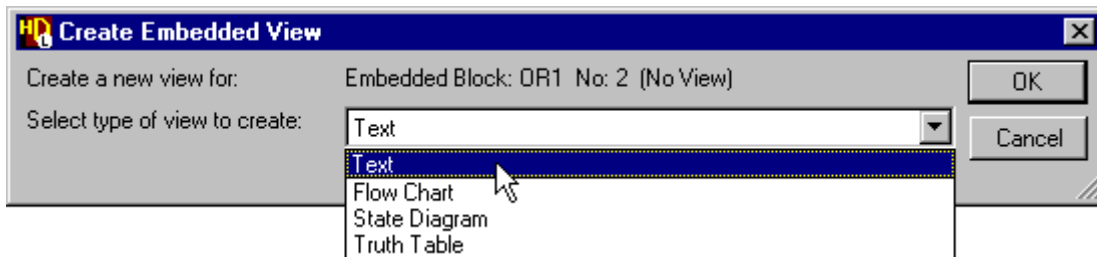
Your block diagram should now look similar to the following picture:




The  button on the dialog box allows you to disclose additional fields which allow you to modify other signal properties including 2D bounds, initial value, kind, attributes and comments. The  button allows you to add additional user-entered architecture declarations to the structural VHDL. Refer to the *HDL Designer Series Graphical Editors User Manual* for more information about these features which are not used in this tutorial.

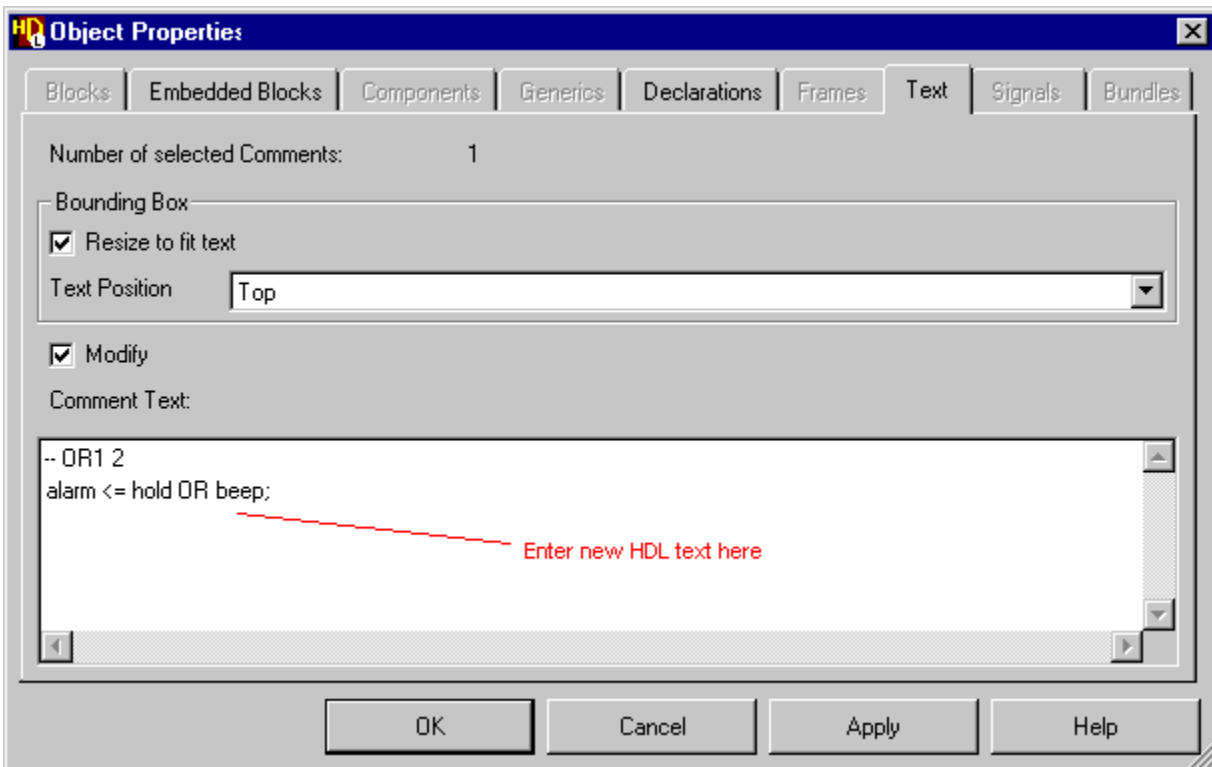
Add an Embedded HDL Text View

30. Select the *OR1* embedded block and display the popup menu by clicking the **Right** mouse button. Choose **New View** from the **Open** cascade in the popup menu to display the Create Embedded View dialog box. Choose Text from the pulldown list of views in the dialog box.



When you confirm the dialog box, an embedded HDL text view containing default text is displayed on the block diagram adjacent to the embedded block.

31. Select the text, re-display the Object Properties dialog box if necessary (using the  button) and choose the **Text** tab.



32. Check the bounding box **Resize to fit text** option and enter the following VHDL statement under the default -- OR1 2 comment text in the dialog box:

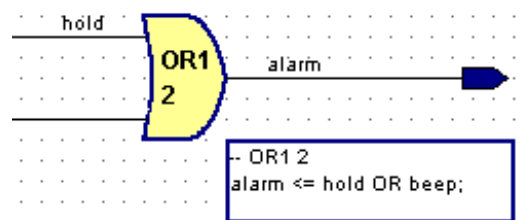
```
alarm <= hold OR beep;
```

The modified HDL text is checked for syntax errors and applied to the diagram when you click the or button on the dialog box.

The functional blocks on the diagram are shown by default as simple rectangular shapes. However, it is sometimes useful to use logic notation when a block has a specific logical function. For example, in this block diagram, the *ORI* embedded block represents a logical OR function. You can apply a logical OR shape using the Comment Graphics toolbar which is normally docked at the bottom of the editor window, by choosing **Autoshapes** from the **Shape** cascade of the popup menu or by using the button in the **Embedded Blocks** tab of the Object Properties dialog box.

33. Select the embedded block and choose the pulldown on the button in the Comment Graphics toolbar to display a palette of alternative shapes. Select from the palette to apply a logical OR shape to the embedded block.
34. Hide the port arrow heads by clearing the **Show Ports when connected** check box in the **Embedded Blocks** tab of the Object Properties dialog box.



The *ORI* embedded block should now look similar to the following picture:



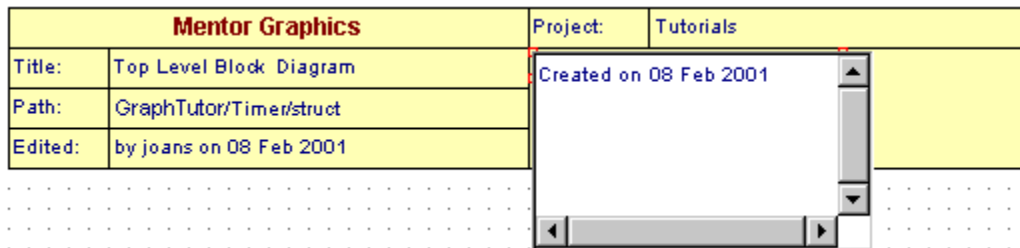
It is also possible to indicate an active low (Not) or edge triggered clock signal. This feature can be used with the alternative shapes to represent extra functions such as an inverter, NAND, NOR or flip-flop. The logical OR function could also be implemented using a ModuleWare part similar to that described in [“Add ModuleWare Components” on page 1-53](#).

Add a Panel and Edit the Title Block


A panel can be useful to outline areas of a diagram. For example, you can use a panel to outline a view used for simulation or animation.

35. Use the  button to add a panel and hold down the  mouse button to drag the panel and enclose the graphical objects on your diagram. The panel is added with the default name *Panel0*.
36. Complete the block diagram by editing the title and comments in the title block on the diagram. For example, enter the title `Top Level Timer Block Diagram` and a comment of the form: `Created by <your name> on <date>`.

The title block comprises a number of grouped comment text objects. Each comment text object can be edited directly by clicking twice on the text to display a text entry box.




You can enter free-format text including line breaks and spaces which will be preserved on the diagram.

37. Click the  mouse button outside the entry box to complete the text entry.



You can also edit an existing comment text object by double-clicking to display the **Text** tab in the Object Properties dialog box. When comments are edited in the dialog box, it is possible to enter any special characters (for example accents or Kanji characters) which are supported on your system.

38. Use the  button to save the block diagram.

You have previously saved the diagram so you are not prompted for library and design unit names. However, you have changed the names of signals connected to input and output ports and the diagram will be inconsistent with the symbol that was automatically created by the previous save. You are prompted whether to update the symbol.

39. Click the button to confirm the save.

The completed block diagram should look similar to the following picture:

Package List

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
```

Declarations

Ports:

```
clk      : std_logic
d        : std_logic_vector(9 DOWNTO 0)
reset    : std_logic
start    : std_logic
stop     : std_logic
alarm    : std_logic
high     : std_logic_vector(3 DOWNTO 0)
low      : std_logic_vector(3 DOWNTO 0)
```

Diagram Signals:

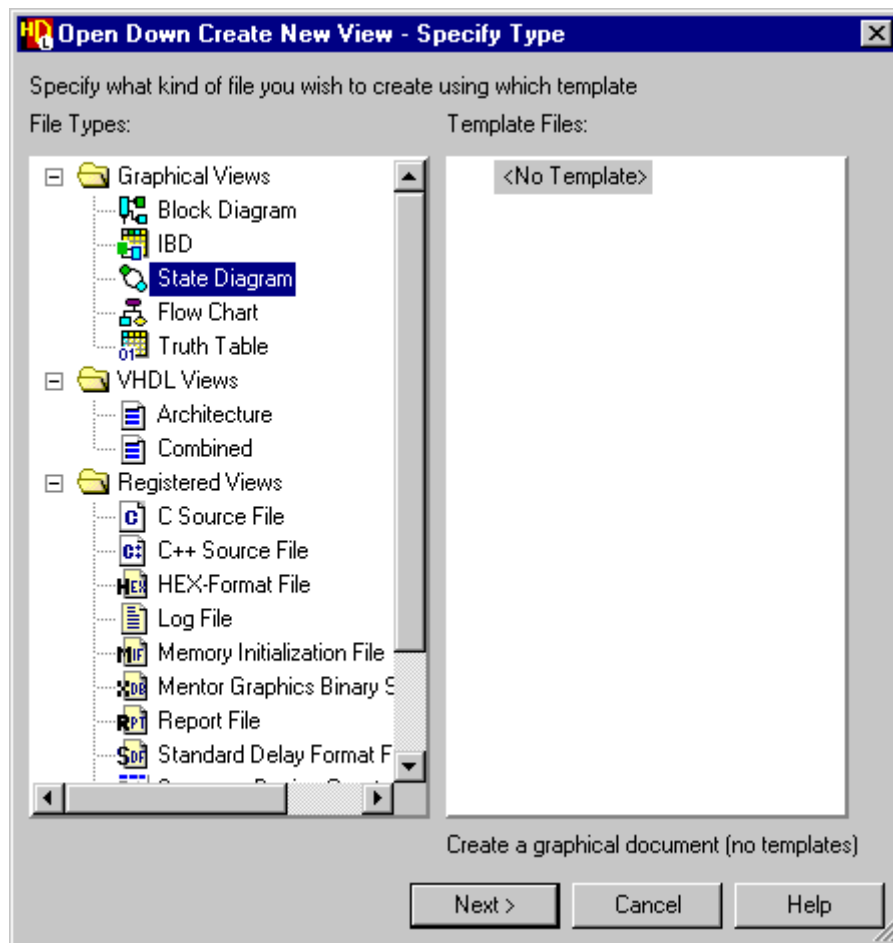
```
SIGNAL beep      : std_logic
SIGNAL clear     : std_logic
SIGNAL dat_in    : std_logic_vector(3 DOWNTO 0)
SIGNAL hold      : std_logic
SIGNAL load      : std_logic
SIGNAL zero      : std_logic
```


Mentor Graphics		Project:	Tutorials
Title:	Top Level Block Diagram	Created on 18th September 2001	
Path:	GraphTutor/Timer/struct		
Edited:	by joans on 18 Sep 2001		

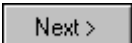
Create a Child State Diagram

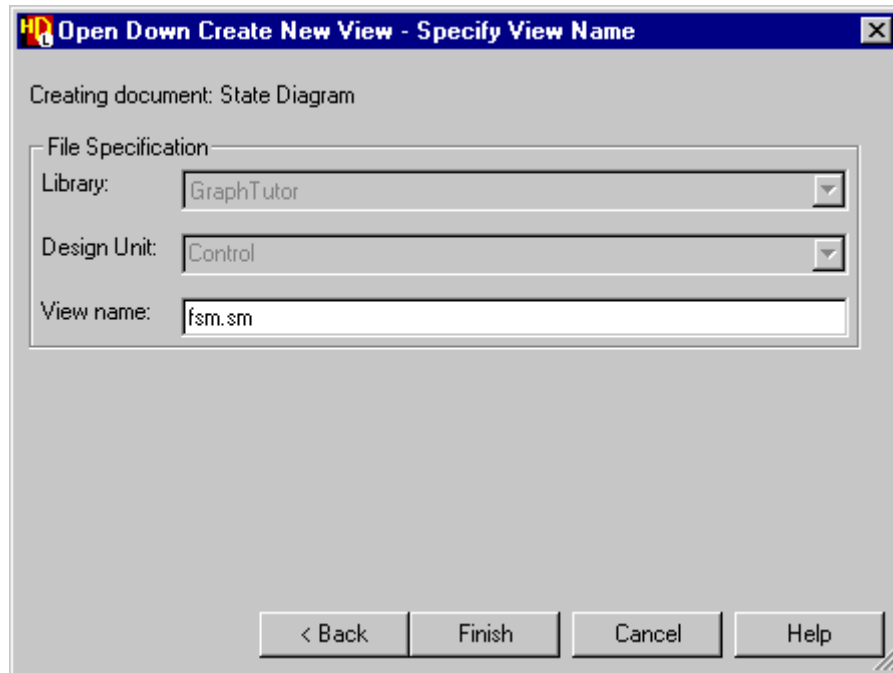
The procedures in this section create a graphical state machine to describe the *Control* block.

1. Move the cursor over the body of the *Control* block on the *Timer* block diagram, then press and release the **Right** mouse button to select the block and display the popup menu.
2. Choose **New View** from the **Open As** cascade menu. The Open Down Create New View wizard is displayed:



Solid handles are displayed when the body of a block (or other re-sizable object) is selected. You can display the wizard directly by double-clicking on the body of a block which has no views defined.


3. Select *State Diagram* from the list of file types and use the  button to display the Specify View Name page of the wizard:



The library and design unit fields are shown dimmed because they are copied automatically from the library and design unit of the parent diagram. The view name defaults to *fsm.sm* for a state machine.



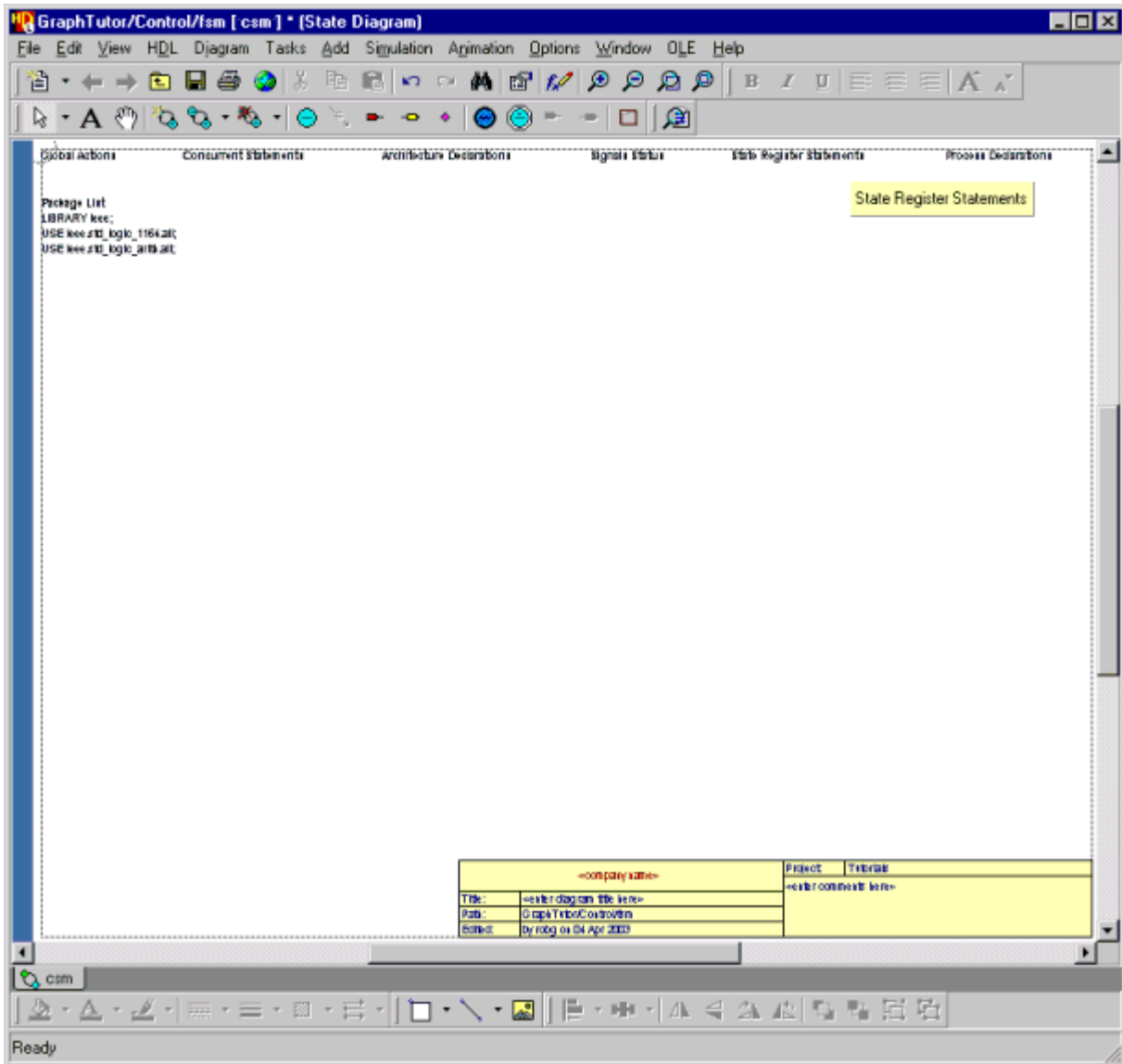
You do not need to enter the file extension (*.bd*, *.fc*, *.ibd*, *.sm* or *.tt*) for graphical views as the correct extension is automatically added. However, if you do enter any other extension you are warned that the file will not be recognized.

4. Use the  button to confirm the wizard with the default view name *fsm.sm*.

A new state diagram *GraphTutor/Control/fsm [csm]* is created as a child view of the *Control* block.


Note that a default state machine name *csm* is appended to the design unit and view names in the diagram title. This name can be changed by choosing **Rename Concurrent State Machine** from the **Diagram** menu in the state diagram.

The state diagram is a blank sheet with page boundaries set for the default printer. The diagram includes text objects for the default VHDL package list and labels for global actions, concurrent statements, architecture declarations, signals status, process declarations and state register statements:




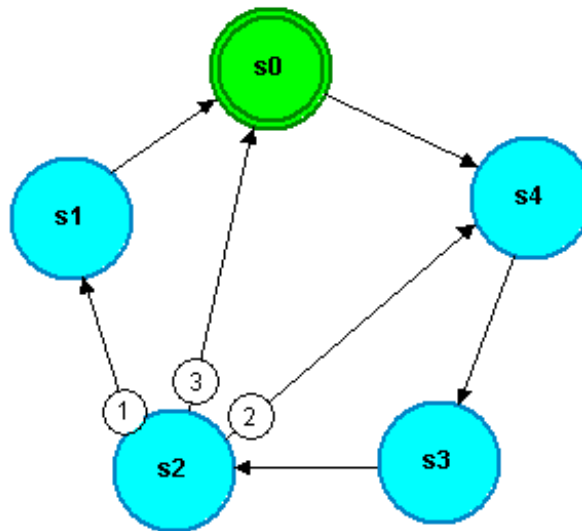
The diagram also includes the default title block which you saved as a template in an earlier topic.

Add States and Transitions

5. Use the  button to add five states on your state diagram. The states are added with default names $s0$, $s1$, $s2$, $s3$ and $s4$.

Notice that the first state you add is assumed to be the start state and is drawn in green with a double outline. The other states are drawn in cyan with a single outline.

6. Use the  button to add transitions between the states as shown in the picture below.




Notice that when you add more than one transition leaving a state, the transition priority is indicated by a number associated with the transition arc. The priorities are initially assigned in the order that you add the transitions but will be re-assigned in a later topic if necessary.






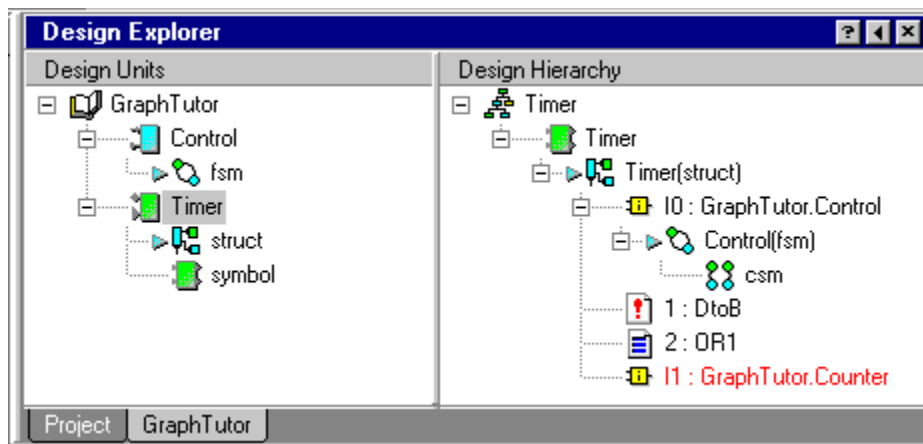
If you add a transition in the wrong direction, you can easily change its direction by choosing **Reverse Direction** from the popup or **Diagram** menu. Note that a popup description (known as an object tip) is displayed when the cursor is stationary over an object. In particular, when the cursor is over a transition, the source state and the destination state are named even if the states are outside the current window.



Save the State Diagram

- Use the  button to save the state diagram. The state diagram was created as a child view from its parent block diagram and is saved using the library, design unit and view names specified when it was created. The design explorer is updated to display the *Control* design unit.

The *Control* design unit is shown as a block in the design explorer because its interface is defined by the connections on its parent block diagram. The *Timer* design unit is shown as a component because it has no parent block diagram and its interface is defined by a symbol.

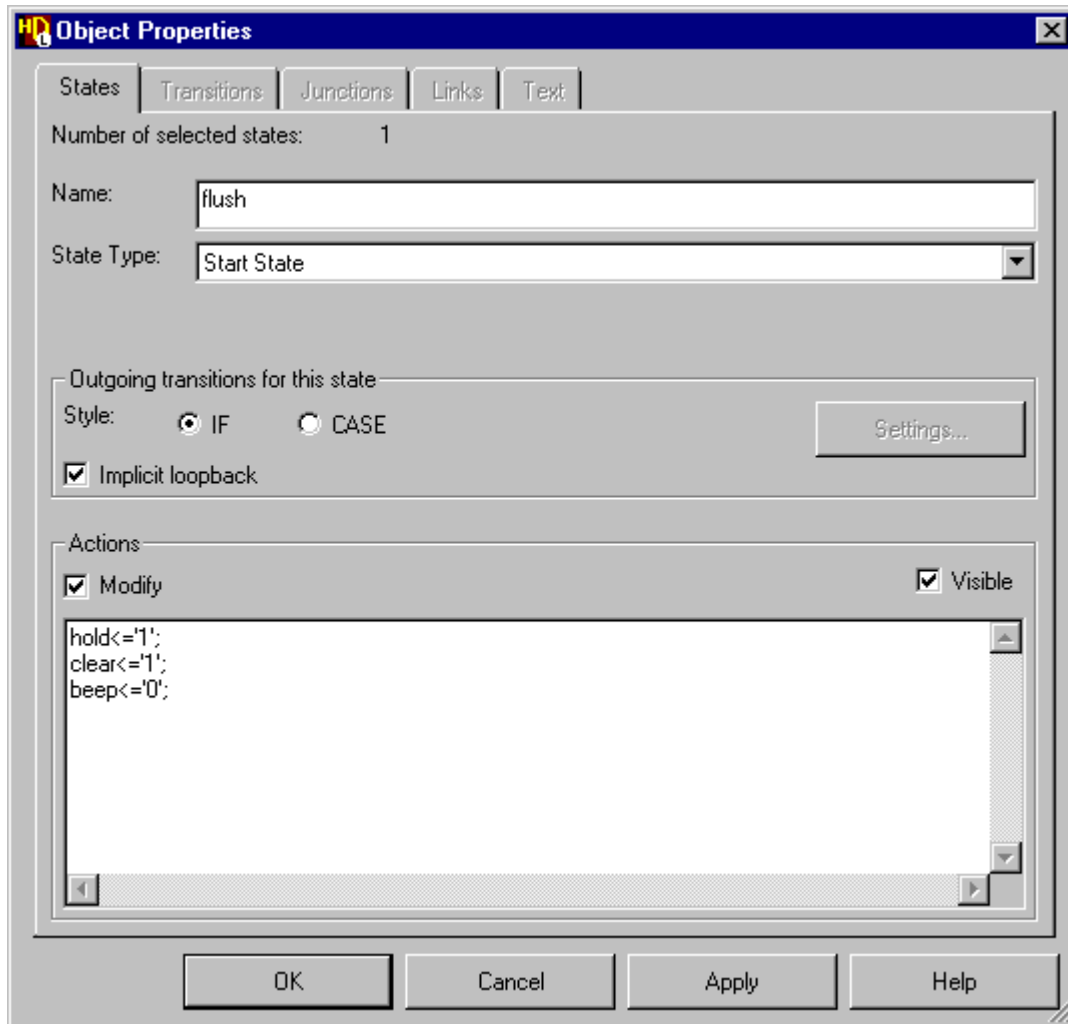
- Click on the  icon for the *Control* design unit in the design explorer to expand the design unit revealing that it contains a state diagram view .
- Select the *Timer* design unit and choose **Show Hierarchy** from the popup menu.
- Select the  Timer hierarchy node in the *Design Hierarchy* pane and choose **Expand All** from the popup menu to reveal the full hierarchy:



Notice that instance *I0* in the hierarchy for the *Timer(struct)* view contains  and  icons showing that the state diagram is described by a concurrent state machine view. The *OR1* embedded block is shown as a text view but the *DtoB* embedded block and the *Counter* instance are not yet defined.

Edit the States

11. Select the start state (*s0* in the picture on [page 1-26](#)) and use the  button to display the **States** tab of the State Machine Object Properties dialog box:





-  You can also display the dialog box by choosing **Object Properties** from the **Edit** menu or by double-clicking on a state.

The **States** tab allows you to enter a name and actions text for one or more selected states on a state diagram. You can also change the visibility of state actions and (when a single state is selected) change the state to a start state or a hierarchical state.

12. Use the dialog box to enter the following state names and actions replacing the default state names *s0* to *s4* in the picture on [page 1-26](#):

Old Name	New Name	Style	Actions
s0	flush	IF	hold<='1'; clear<='1'; beep<='0';
s1	count	IF	(no actions)
s2	getkey	IF	hold<='1';
s3	load_t	IF	hold<='1';
s4	load_u	IF	hold<='1'; load<='1';

The VHDL Expression Builder dialog box is automatically displayed when you start to enter the actions. The dialog box can be used to choose from lists of the available port or local signal names, operators and example values. For example, choose the *hold* signal,  button, value '1' and  button to enter the action *hold* <= '1';.



The Expression Builder dialog box can be explicitly displayed at any time by choosing **Expression Builder** from the **Edit** menu.

The syntax for state actions is automatically checked and any errors reported on entry. VHDL statements must be terminated by a semicolon (;) character. Line breaks and spaces can be used for clarity and will be preserved on the diagram.



You can also edit the state name or actions by direct text editing on the diagram or copy a state and paste its state actions into another state by choosing the **Paste State Actions** option from the **Paste Special** cascade in the popup menu.

If the name is larger than the state, it auto-resizes to fit the new name. You can also resize a state by selecting the state and dragging one of its resize handles but you cannot make it smaller than the enclosed name.



Edit the Transitions



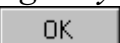
13. Add conditions to the transitions as shown in the following table:

Origin	Destination	Priority	Condition	Actions
count	flush	1	stop='1'	none
getkey	flush	3	stop='1'	none
getkey	count	1	start='1'	none
getkey	load_u	2	d/=NOUGHTS	none
flush	load_u	1	d/=NOUGHTS	none
load_u	load_t	1	(none)	none
load_t	getkey	1	d/=NOUGHTS	none





The *NOUGHTS* text string is the name of a constant which will be declared as a state machine property later in this tutorial.

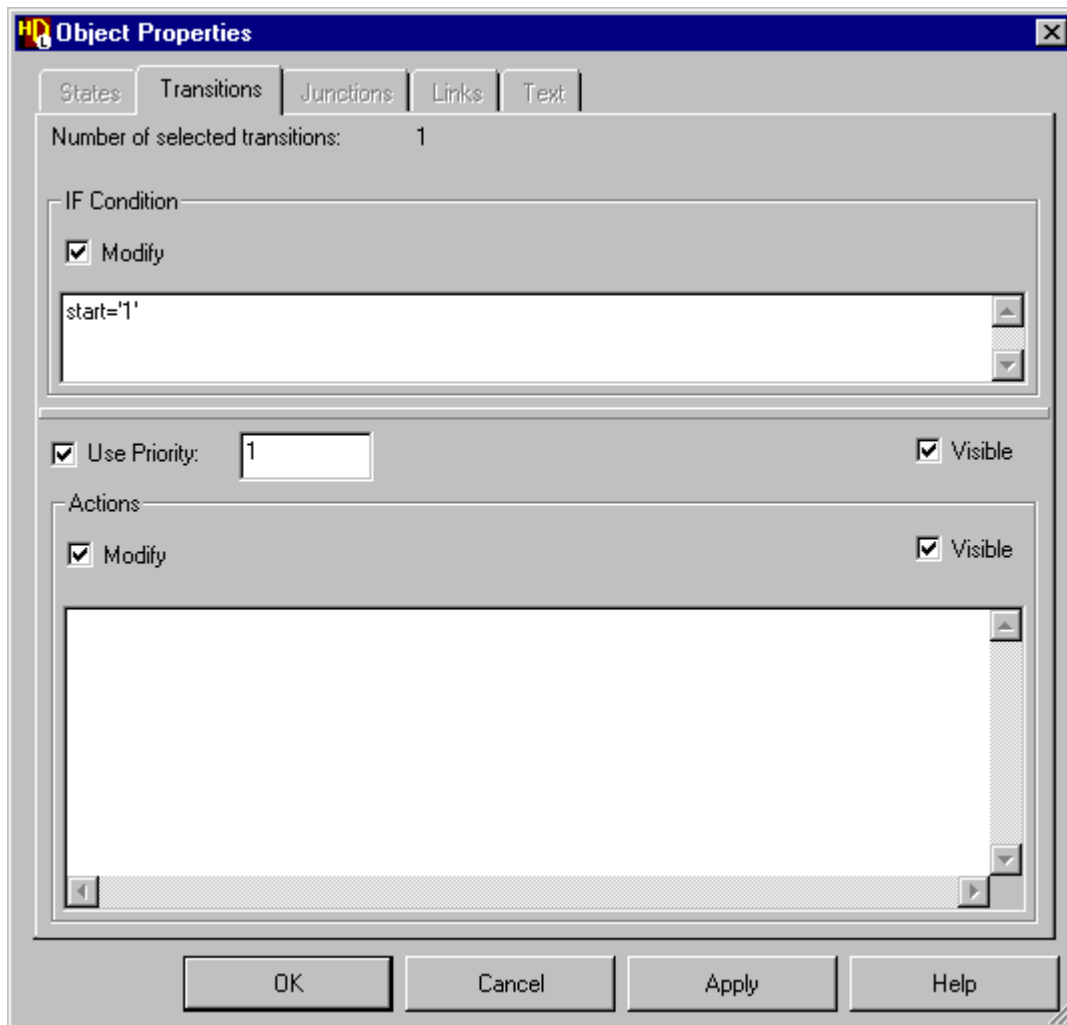
Hold down the  mouse button and select the two transitions entering the state *flush* by dragging the cursor across them (or  mouse button).

Use the  button to display the **Transitions** tab of the State Machine Object Properties dialog box. Enter the condition text *stop='1'* in the dialog box using the expression builder and click the  button to add this condition to both of the selected transitions. A confirmation dialog box is displayed warning that the transition from state *getkey* to state *count* has no condition and is not the lowest priority. Click  to acknowledge the warning. Repeat this procedure for the *start='1'* and *d/=NOUGHTS* conditions.


The condition syntax is checked on entry. Any valid VHDL fragment can be entered using line breaks and indentation to improve the legibility on the diagram if required.

Ensure that the transition priorities leaving the state *getkey* are the same as those shown in the table. You can change a transition priority in the **Transitions** tab of the State Machine Object Properties dialog box. Alternatively, you can use the  button to zoom in or the  button to view an area and use direct text editing to change the priority.

When you change a transition priority, the priority of the other transitions leaving the same state are automatically adjusted.

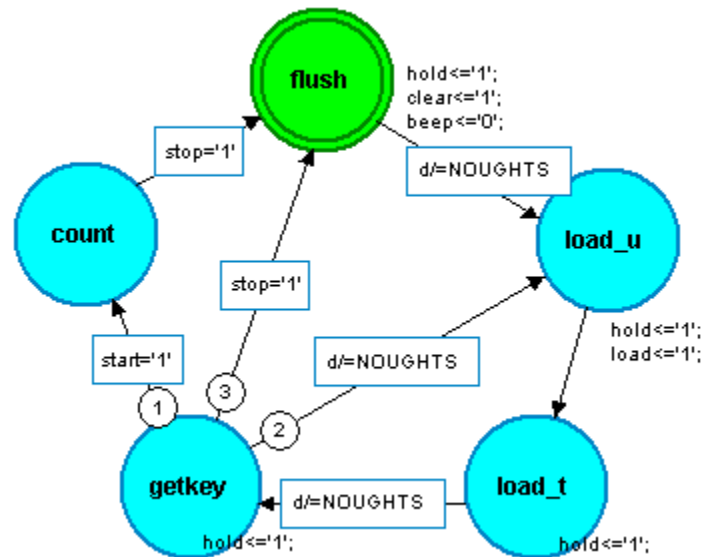


If you have zoomed in, use the  button to view all of the state diagram.

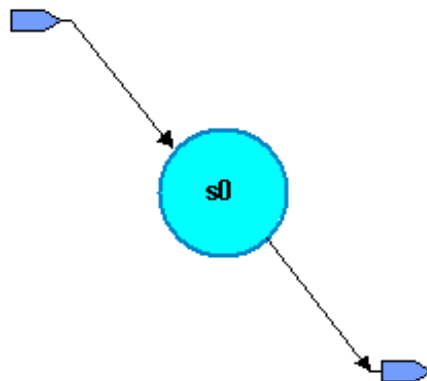
14. Use the  button to save your changes to the state diagram.

Create a Hierarchical State Diagram




The state diagram should look similar to the following picture.

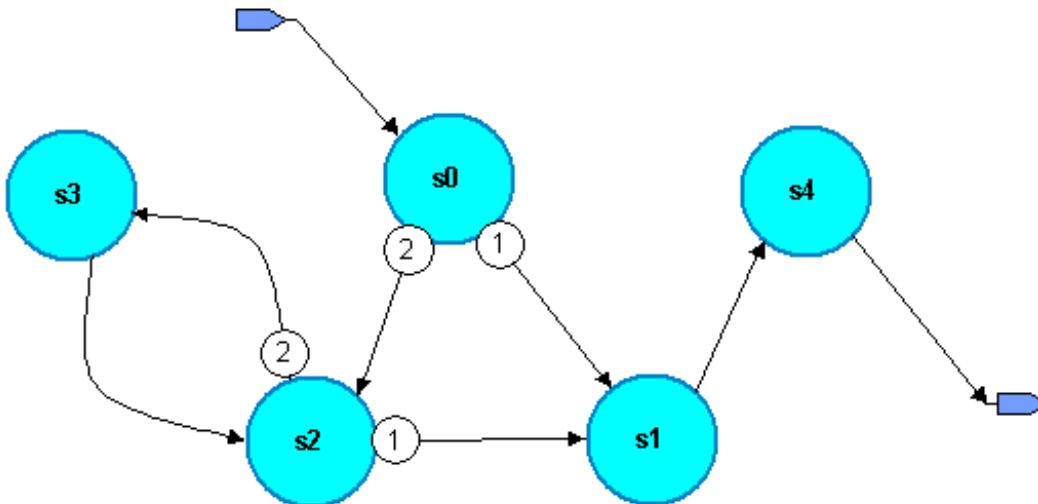


15. Select the *count* state and choose **Hierarchical State** in the **Change To** cascade from the **Diagram** menu. Alternatively, choose the Hierarchical State pulldown option for the State Type in the **States** tab of the State Machine Object Properties dialog box. The *count* state is redrawn as a hierarchical state with a triple outline and darker fill color.
16. Double-click on the hierarchical state (or use the **Open Down** option from the popup menu) to create the new hierarchical child state diagram. A new child state diagram is opened as a tabbed window initialized with a default state *s0* connected to an entry point and exit point.



Notice that the name of the hierarchical state *count* is included in the window title: *GraphTutor/Control/fsm [csm/count]*. This convention shows that the child diagram is a partial view of the parent diagram.

17. Select the exit point and choose **State** in the **Change to cascade** of the **Diagram** menu.
18. While the new state is selected, drag with the  mouse button and release the mouse button with the ghosted state to the left and below the first state. Use the **Copy Here** option from the popup menu to make a copy of the state at the cursor position.
19. Repeat this procedure to add two more states on the diagram. This method for adding objects can be useful when you want to add an object with the same or similar properties and attributes to an existing object.
20. Use the  button to add a new exit point and the  button to connect transitions between the states as shown in the following picture:



You can add route points by clicking at several points between states to create a smooth arc as shown in the picture between states *s2* and *s3*. You are not limited by the page boundaries when you add objects or edit a graphical diagram. If necessary, you can drag a rectangle around the objects to select them and move them back inside the page boundary or choose **Refresh Page Boundaries** from the **File** menu to show how the edited diagram will be fitted onto printer pages.

Complete the Hierarchical State Diagram

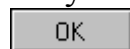
21. Use the **States** tab of the State Machine Object Properties dialog box to rename the states and add state actions as shown below:

Old Name	New Name	Style	Actions
s0	standby	IF	hold<='1';
s1	alarm	IF	hold<='1'; clear<='1'; beep<='1';
s2	counting	IF	(no actions)
s3	suspended	IF	hold<='1';
s4	end_count	IF	hold<='1'; clear<='1';


22. Use the **Transitions** tab to add the following conditions:

Origin State	Destination State	Priority	Condition	Actions
suspended	counting	1	stop='0'	none
counting	suspended	2	stop='1'	none
counting	alarm	1	zero='1'	none
standby	counting	2	start='1'	none
standby	alarm	1	zero='1'	none
alarm	end_count	1	stop='1'	none

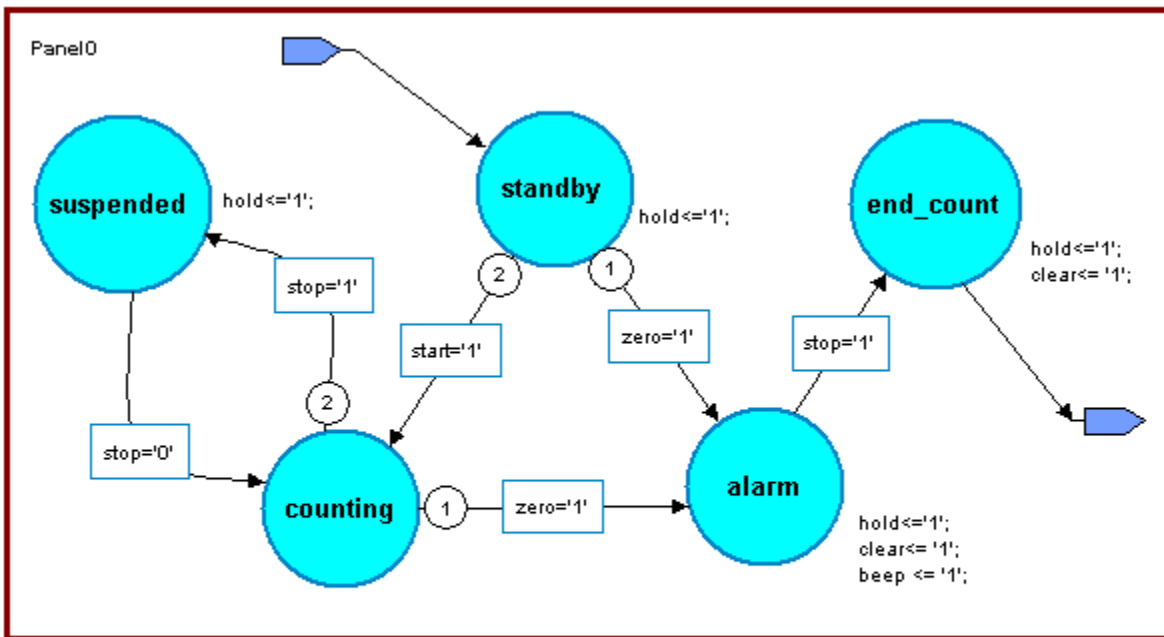
A confirmation box may be displayed warning that a transition you have not yet edited has no condition and is not the lowest priority. Click the



button to acknowledge the warning.


23. Complete the hierarchical state diagram by editing the title and comments in the title block and using the  button to add a panel around the graphical objects on your diagram. This panel can be used to set the diagram view when you animate the state diagram later in this tutorial.

The state diagram should look similar to the picture below:



Mentor Graphics		Project:	Tutorials
Title:	Counter state machine	Child hierarchical view of the count state in the Counter state machine.	
Path:	GraphTutor/Control/fsm		
Edited:	by joans on 04 Apr 2003		



i Ensure that the transition priorities are as shown and that you include the terminating semi-colon when you enter the state actions.


24. Use the  button to save the state diagram.

Although it is displayed in a separate tab, a child hierarchical diagram is a partial view of a state machine and the parent and child diagrams are saved as a single design unit view (*GraphTutor/Control/fsm.sm*).

Editing State Machine Properties

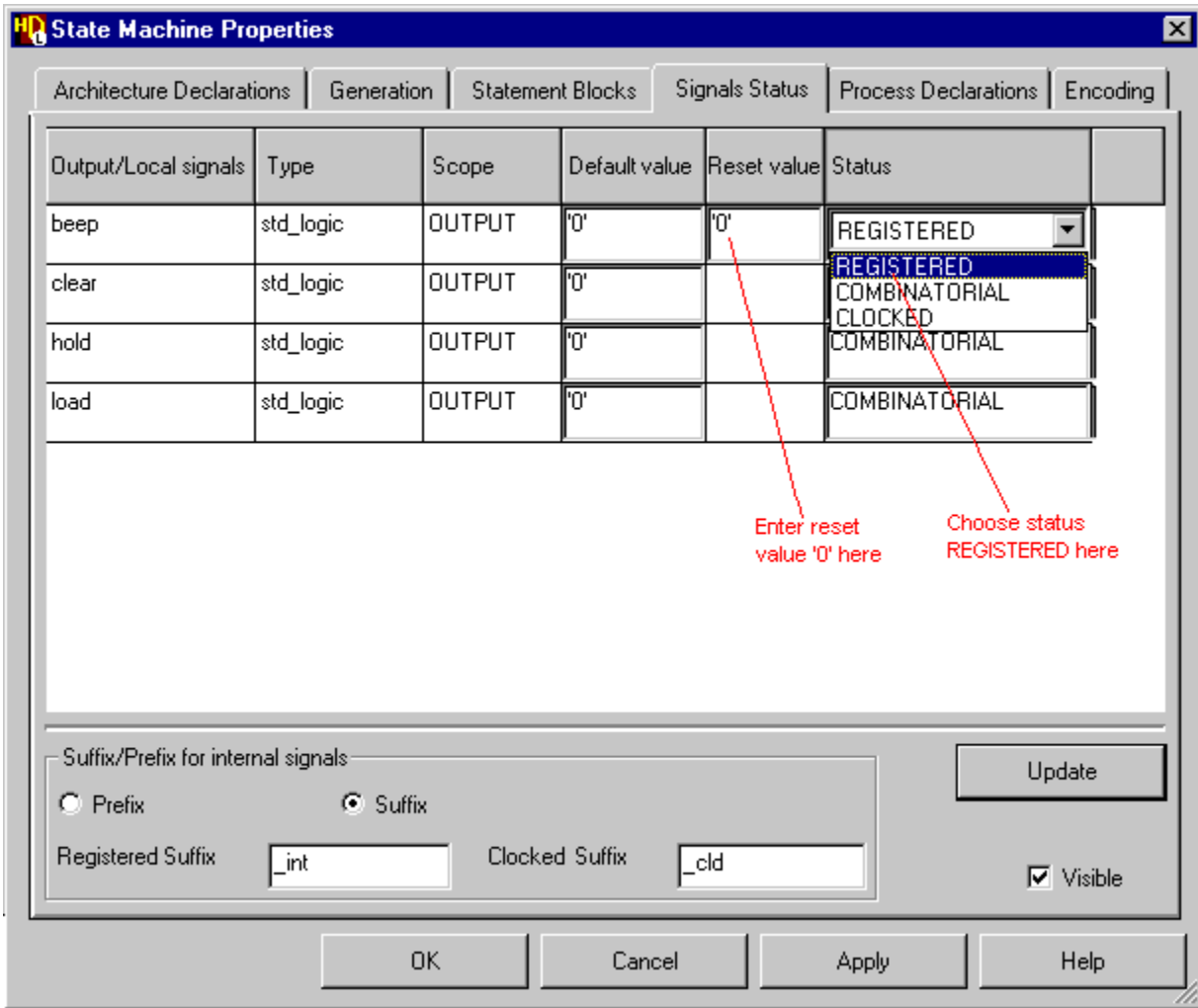
There are a number of properties associated with a state machine which can be edited using the State Machine Properties dialog box. The dialog box can be accessed by choosing **State Machine Properties** from the **Diagram** or popup menu in the parent or child state diagram. It can also be accessed by double-clicking over one of the labels which are displayed in the parent diagram of a hierarchical state machine.

25. Use the  button (or choose the **Open Up** option from the **File** menu) to re-display the parent state diagram if it is not already displayed.
26. Double-click the  mouse button over the Global Actions, Concurrent Statements or State Registers label on the diagram to display the **Statement Blocks** tab of the State Machine Properties dialog box. There are no global actions, concurrent statements or state register statements required in this design. Hide the labels for these objects on the state diagram by clearing the **Visible** check box for Global Actions, Concurrent Statements and State Register Statements in the **Statement Blocks** tab.
27. Choose the **Process Declarations** tab. As there are no process declarations required in this design, hide this label by clearing the **Visible** check box in the **Process Declarations** tab.
28. Choose the **Architecture Declarations** tab and enter the following declaration for the 10-bit constant NOUGHTS in the free-format entry box:

```
Constant NOUGHTS : std_logic_vector := "0000000000";
```
29. Click the  button to confirm the dialog box and update the diagram. The constant declaration is added below the Architecture Declarations label on the diagram and will be included as architecture declarations when HDL is generated for the state machine. All other labels other than the default signals status should now be hidden on the diagram.
30. Re-display the State Machine Properties dialog box and choose the **Signals Status** tab by double-clicking over the Signals Status label on the diagram.

Notice how the output signals are listed in the dialog box with the default value set to '0' and the status *COMBINATORIAL*.


31. Click on the Status field for the *beep* signal and choose *REGISTERED* from the drop down box.
32. Click in the Reset value field and enter the value '0'.



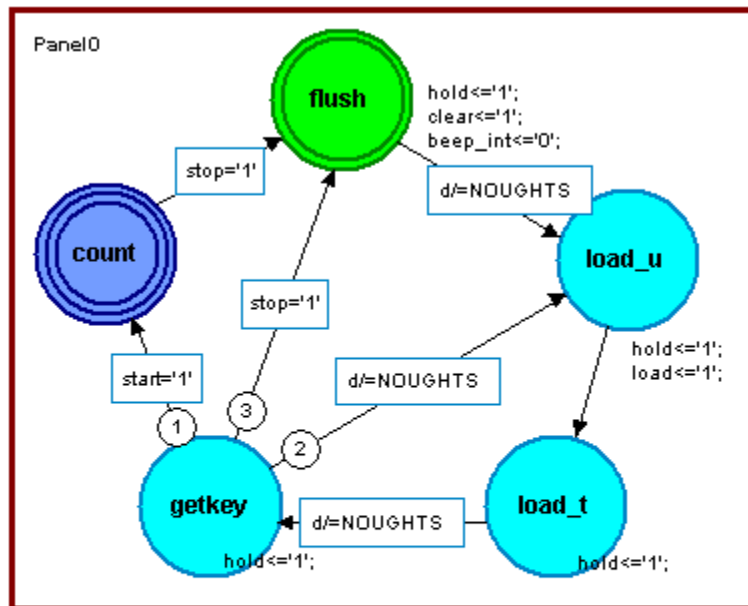
Any locally declared internal signals would be shown with the status REGISTERED. Bidirectional or buffer signals are treated as output signals.

The dialog box also allows you to change the suffix or prefix used for internal registered or clocked signal names. For this tutorial, suffixes are used with the default values *_int* and *_cld*.

33. Confirm the dialog box by clicking the  button.

34. Complete the state diagram by editing the title and comments in the title block and using the  button to add a panel around the graphical objects on your diagram. This panel can be used to set the diagram view when you animate the state diagram later in this tutorial.

The final state diagram should look similar to the picture below:




```

Architecture Declarations
Constant NOUGHTS : std_logic_vector := "0000000000";

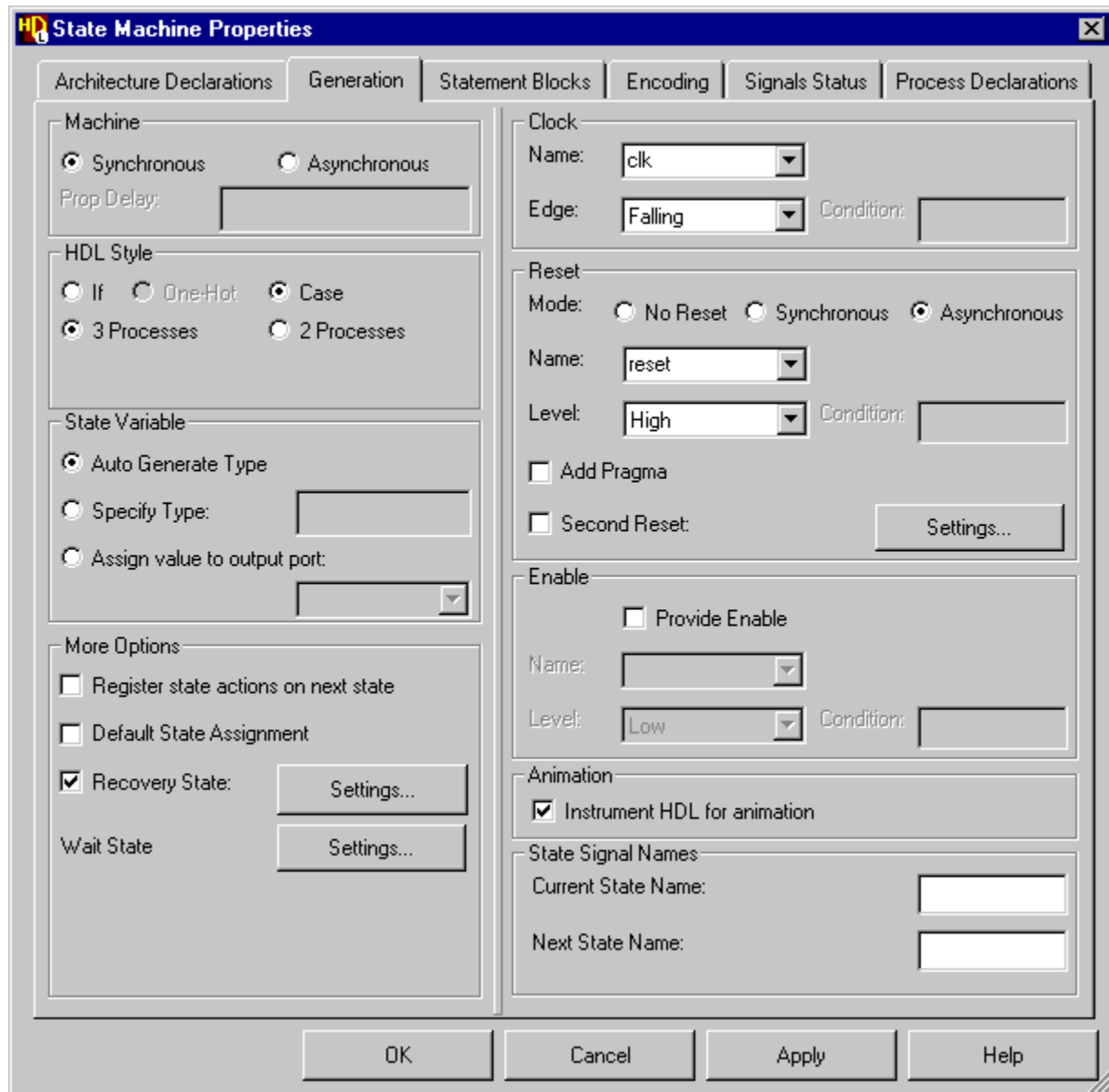
Signals Status
SIGNAL SCOPE DEFAULT RESET STATUS Package List
beep OUT '0' '0' REG LIBRARY ieee;
clear OUT '0' COMB USE ieee.std_logic_1164.all;
hold OUT '0' COMB USE ieee.std_logic_1164.all;
load OUT '0' COMB USE ieee.std_logic_arith.all;
    
```

Notice that all occurrences of the *beep* signal (in the *flush* state and in the *alarm* state on the child hierarchical state diagram) have been replaced by the internal signal name *beep_int* using the default suffix *_int*.


The State machine Properties dialog box also provides tabs for setting HDL generation characteristics and state machine encoding. State machine encoding is not used in this tutorial and the encoding scheme should be set to **None**. You can use the  buttons for more information about each tab of the State Machine Properties dialog box.

Set Generation Properties

35. Re-display the State Machine Properties dialog box if necessary and choose the **Generation** tab. This tab sets properties used for HDL generation.



36. Choose the **Synchronous** option in the Machine box. Use the default options **Case** and **3 Processes** for HDL Style and **Auto Generate Type** for the State Variable.





37. Check that the **Recovery State** is set to `<start_state>` by using the  button to display the Recovery State Settings dialog box.



This entry automatically assigns the start state or you can choose from a pull-down list of states and specify recovery state actions.

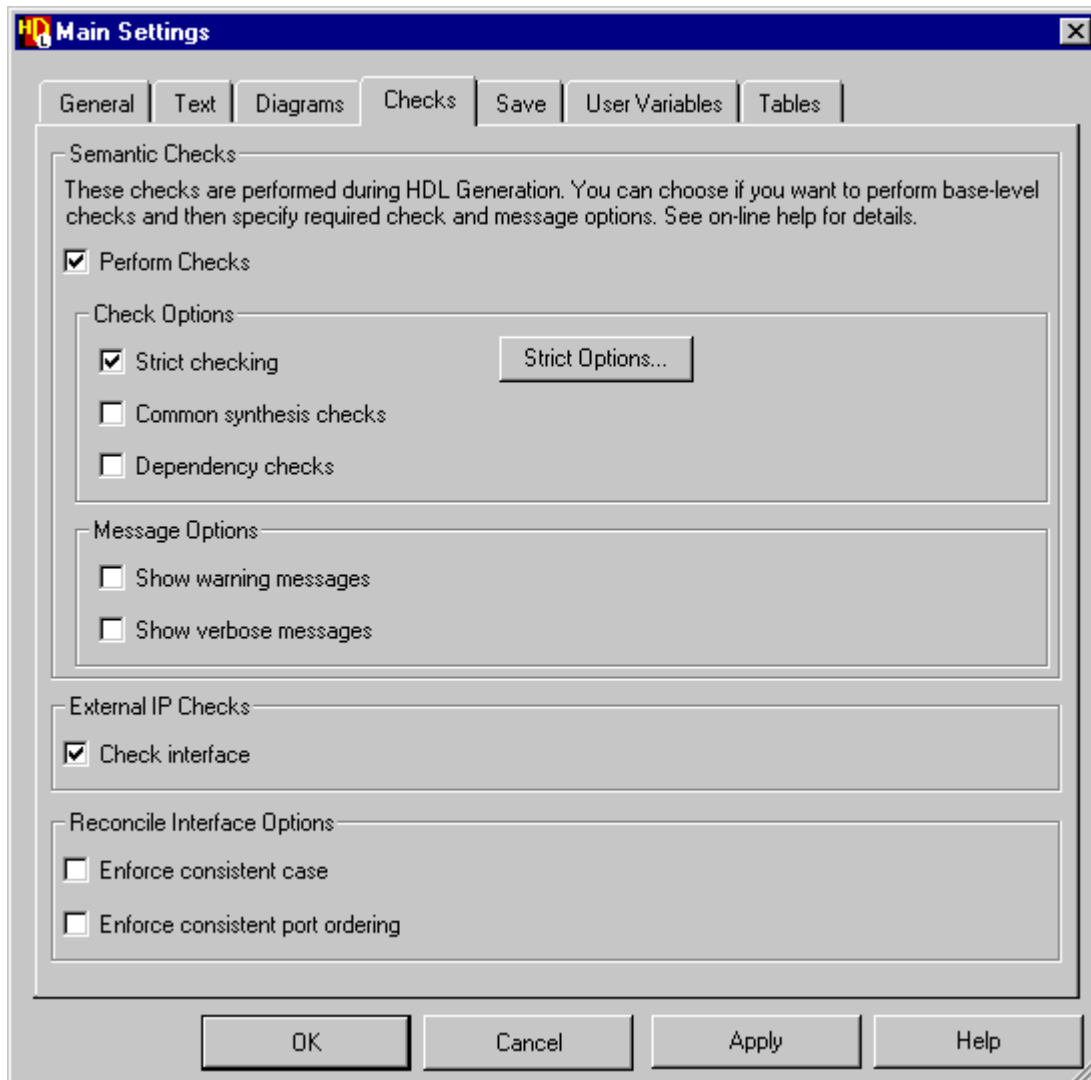
38. Leave the **Register state actions on next state** and **Default State Assignment** options unset.
39. Choose *clk* from the pull-down list of clock signal names and *Falling* from the Edge pull-down list. Set an **Asynchronous** reset and choose the signal name *reset* with a *High* level from the pull-down lists. Leave the **Provide Enable** options unset.
40. If a ModelSim or NC-Sim simulator is available, set the **Instrument HDL for animation** option. This option adds additional code to the generated HDL which is used for state machine animation later in this tutorial.

 This additional code is surrounded by translation control pragmas which ensure that it is ignored by downstream synthesis tools.

41. Use the  button (or  followed by ) to apply the generation properties to your state machine and dismiss the dialog box.
42. Use the  button to save the state diagram.

Set Checks for HDL Generation

You can set the level of checks performed when HDL is generated by using the **Checks** tab in the Main Settings dialog box which can be accessed by choosing **Main** from the **Options** menu.




For this tutorial, the **Strict checking** option should be selected.



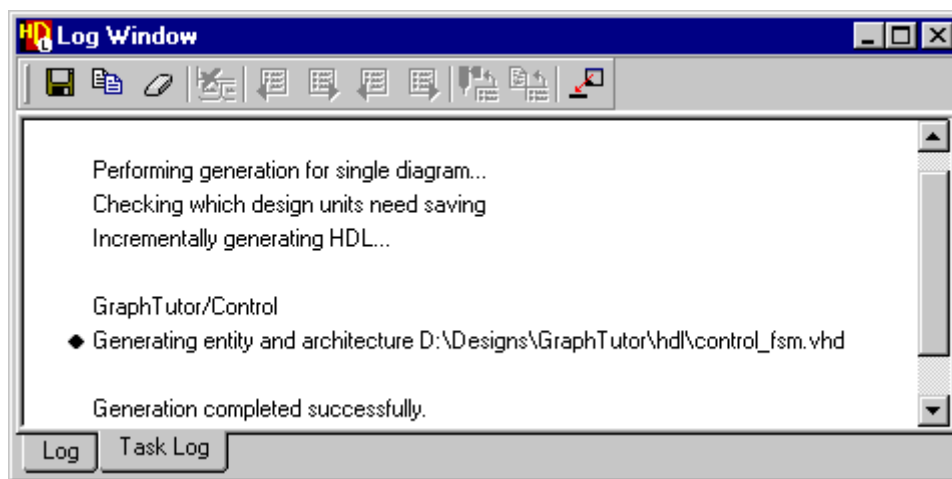
The **Strict Options...** button can be used to display a dialog box which allows you to modify the strict checking options. The default strict check options should be used for this tutorial.

Generate HDL for the State Machine





Although the *Timer* design is not yet complete, you can now generate HDL for the *Control* state machine.



43. Choose the **Run Single** option from the **Generate** cascade in the **Tasks** menu (or use the  button) in the state diagram window.


The progress of the HDL generation is monitored in a log window which includes any errors and warnings issued during generation.





This example shows a combined file is generated for the VHDL entity and VHDL architecture body. Separate files may be generated if you have changed the preferences for VHDL file options.

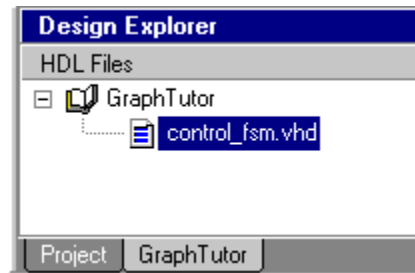
If there are any errors, you can move to the next error message using the  button or the previous error using the  button. You can display the source graphics corresponding to the error by double-clicking on the error message (or using the  button when the error is selected). Alternatively, you can use the  button when the error is selected to view the generated HDL. If your text editor supports a line number argument, the HDL line corresponding to the error is selected automatically.


If there are no errors, you can use the toolbar buttons  and  to view the source graphics or generated HDL corresponding to any message which is marked by the ◆ icon in the log window. For example, the “Generating entity and architecture ...” message shown above.

Generated HDL files can also be opened by using the  button in any graphics editor window or when the diagram view is selected in the design explorer.


Display HDL Files in the Design Explorer

44. Use the  button (or choose HDL Files from the **Mode** cascade in the **View** menu) to display the *HDL Files* view in the design explorer.
45. Click on the  icon for the *GraphTutor* library and notice that the view is expanded to reveal the generated VHDL architecture body file for the *Control* design unit.



 If you have changed your preferences to create separate VHDL entity and architecture body files, then both files are shown.

46. Open the HDL file by double-clicking on the generated file name or by choosing **Open File** from the popup menu.



If you are using the DesignPad HDL text editor, you can use the  button or the **Trace To Graphics** command from the **Document** menu in DesignPad to cross-reference from a line in the generated HDL to the corresponding graphics object.

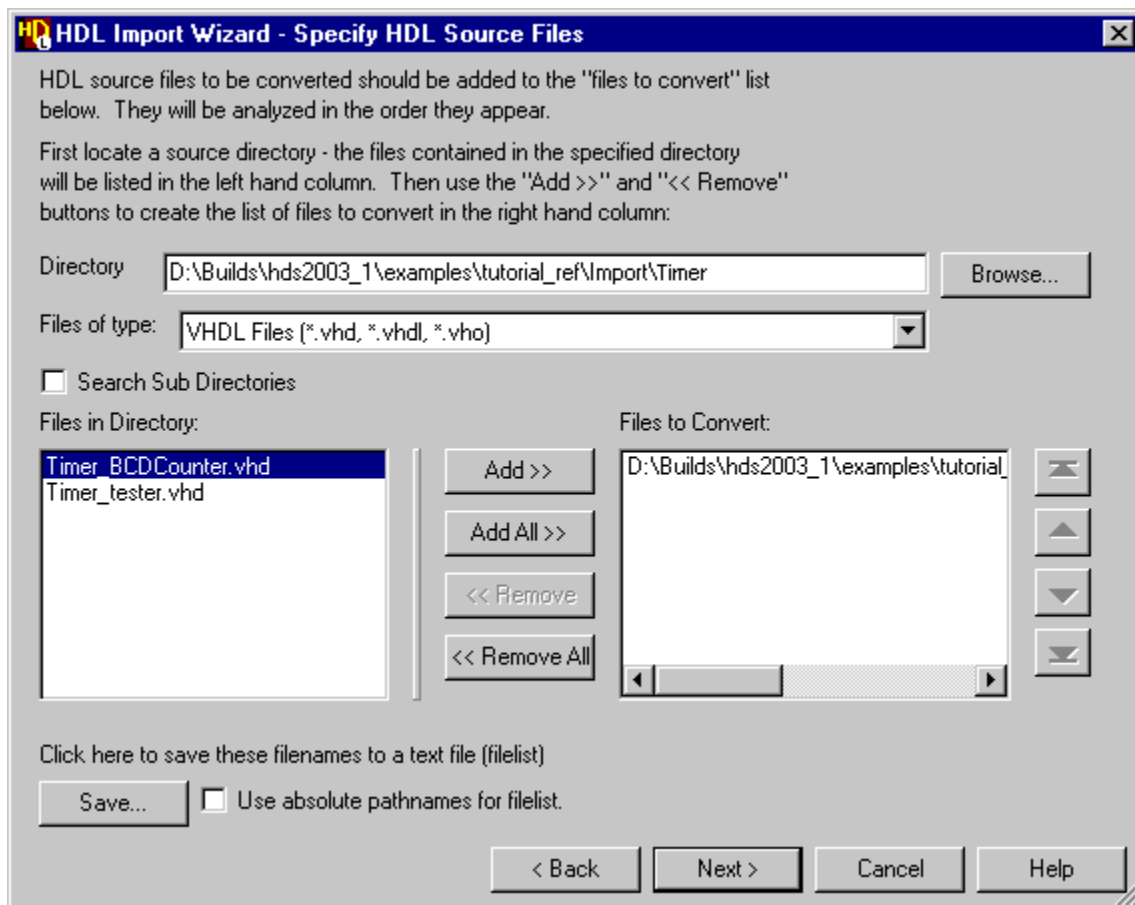
A similar option can be set up for other user customizable external text editors. Refer to the “Text Editors” chapter in the *HDL Designer Series User Manual* for more information.



47. Close the text editor after you have viewed the generated HDL.
48. Use the **Close Window** option from the **File** menu to close the parent *GraphTutor/Control/fsm [csm]* and the child *GraphTutor/Control/fsm [csm/count]* views of the state machine.



Import the BCDCounter Design Unit

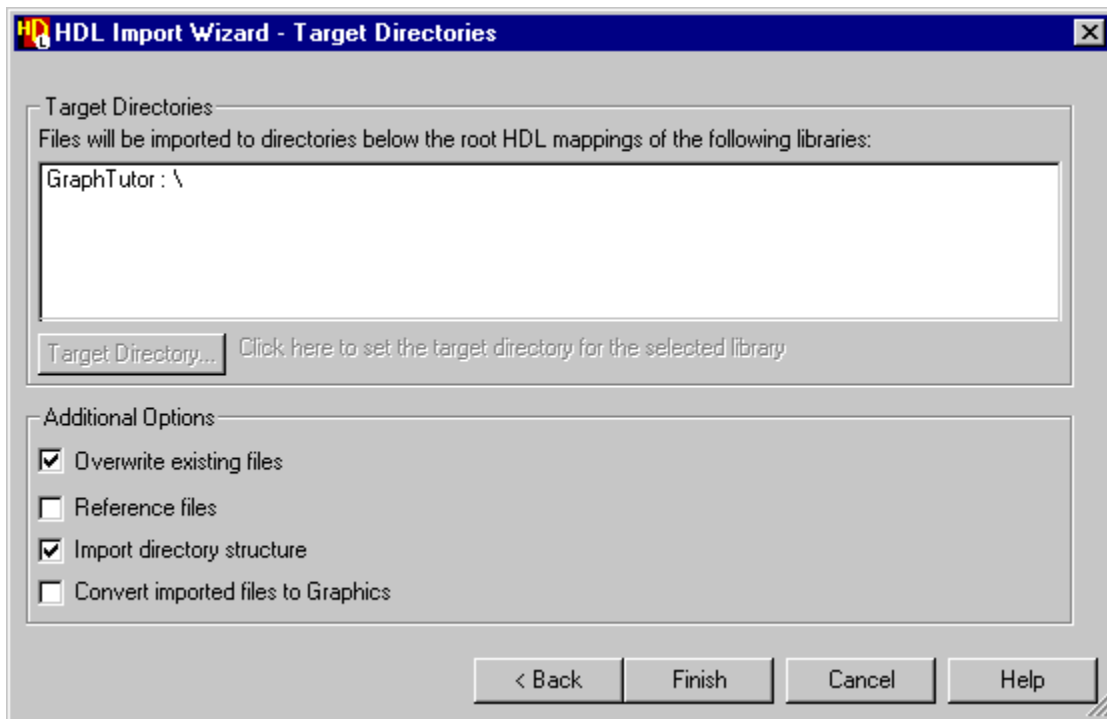
You have now defined the *Control* block by drawing a graphical state machine. The *Counter* block will be defined by instantiating two instances of a HDL text component in a child block diagram.


1. Use the  button in the design manager window (or choose **HDL Import** from the **HDL** menu) to display the HDL Import wizard.
2. Choose **Specify HDL files** and **VHDL** in the first page of the wizard and click the  button to display the Specify HDL Source Files page.



3. Choose VHDL files from the pulldown filter for Files of type and use the  button to locate the `..\examples\tutorial_ref\Import\Timer` installation subdirectory. Select the `Timer_BCDCounter.vhd` file and add it to the list of files for conversion by using the  button.

4. Click the  button on the wizard to display the Target Libraries page. This page allows you to select the default target library and add libraries if there are any instantiations for black box components in the source HDL code.
5. Ensure that your *GraphTutor* library is selected as the target library and click the  button to display the Target Directories page. This page allows you to control the imported directory structure and set additional HDL import options:



6. Click the  button in this page to complete the HDL import using the default target directory and options specified in the wizard. The following completion message should be displayed in the HDL log window:

```
** Importing 'D:\Designs\GraphTutor\hdl\Timer_BCDCounter.vhd'
HDL Import complete
```

```
1 file imported to 1 library
```

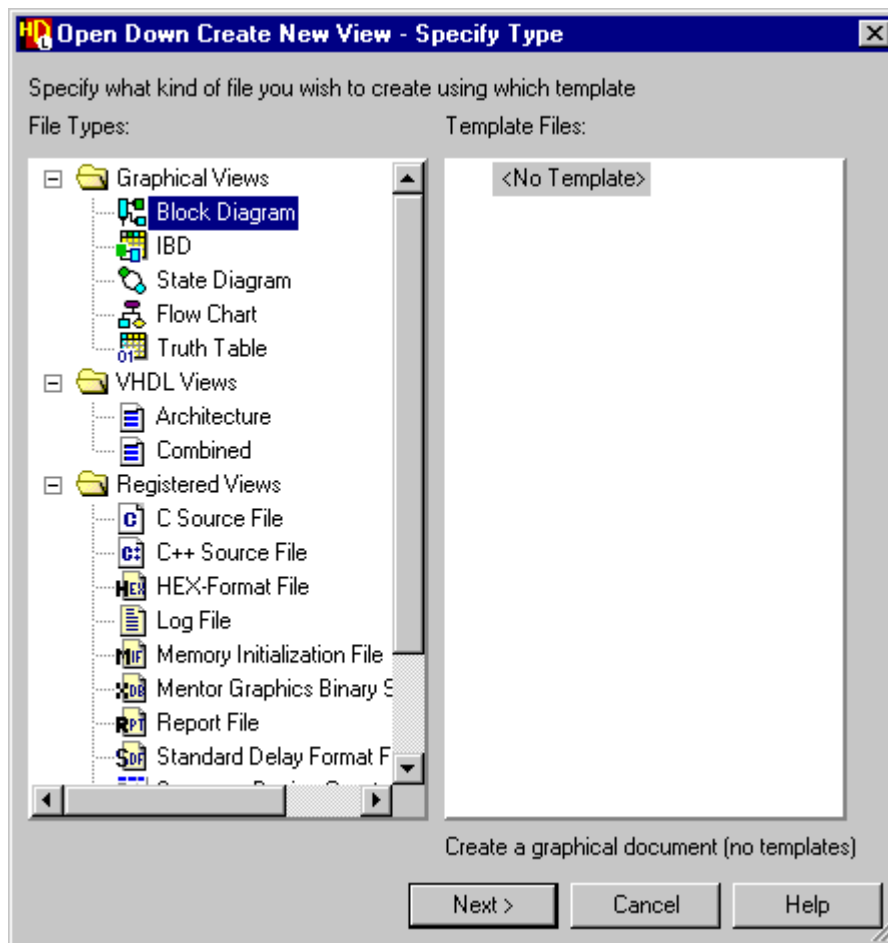
Create a Child Block Diagram


1. Display the *Timer* block diagram.




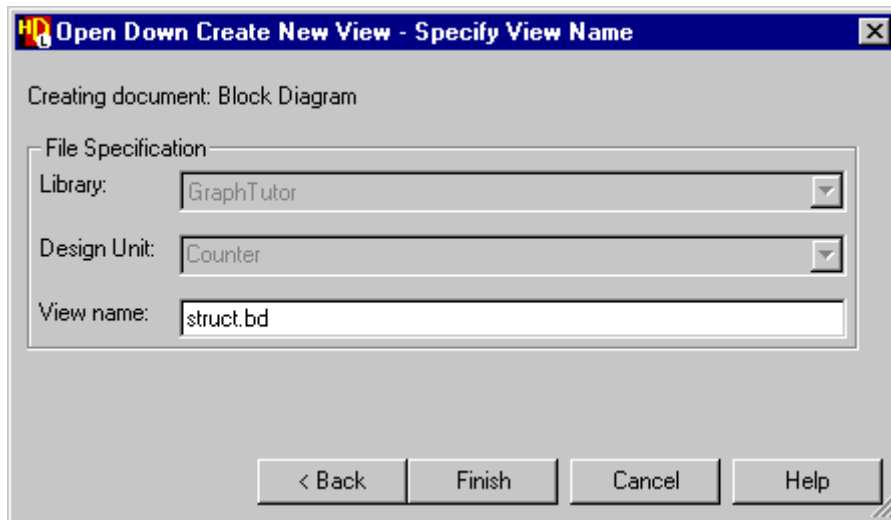
If the window is obscured, you can pop it to the front by selecting the window name from the **Windows** menu list in any other window.

2. Double-click on the body of the *Counter* block (or choose New View from the **Open As** cascade of the popup menu) to display the Open Down Create New View dialog box.



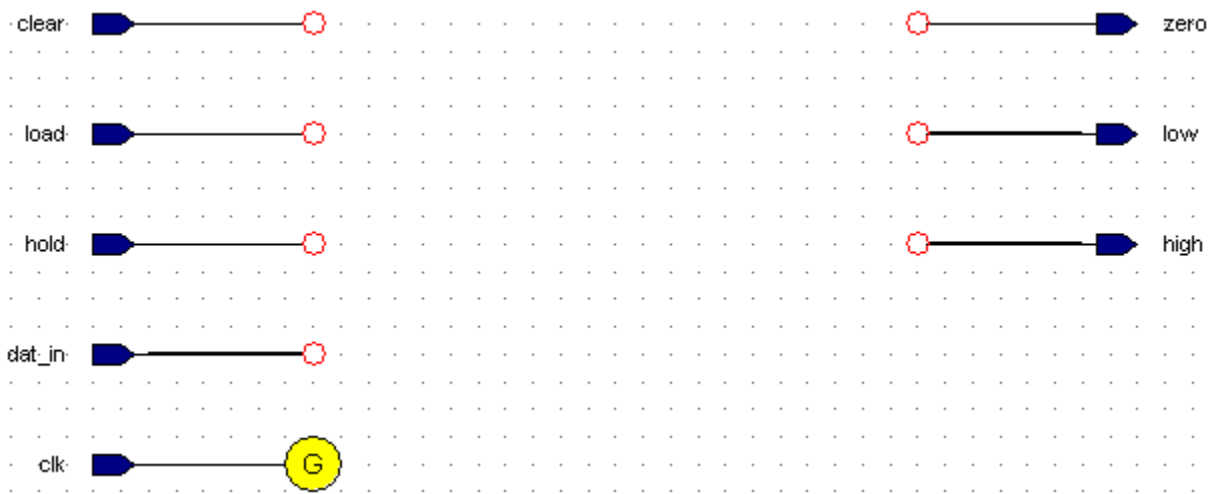
3. Select *Block Diagram* from the list of file types and use the  button to display the Specify View Name page of the wizard.

4. Use the  button to confirm the wizard with the default view name *struct.bd*.



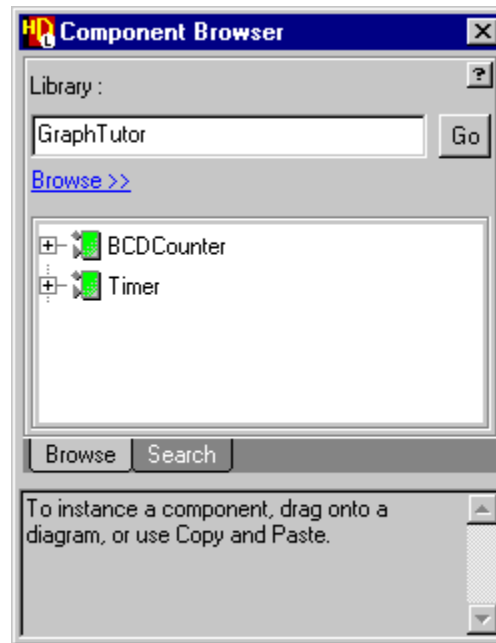
A new block diagram *GraphTutor/Counter/struct* is created as a child view of the *Counter* block. The block diagram is initialized with declarations and ports corresponding to the interface signals and buses on the parent diagram (including a global connector for the *clk* signal).

Notice how inputs are initialized on the left and outputs on the right.

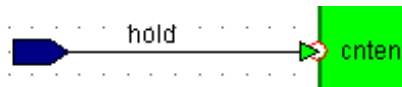



5. Use the  button (or choose **Component** from the **Add** menu) to display the Component Browser.



- Click on [Browse >>](#) to display a drop down list of libraries and double-click to choose your *GraphTutor* library:

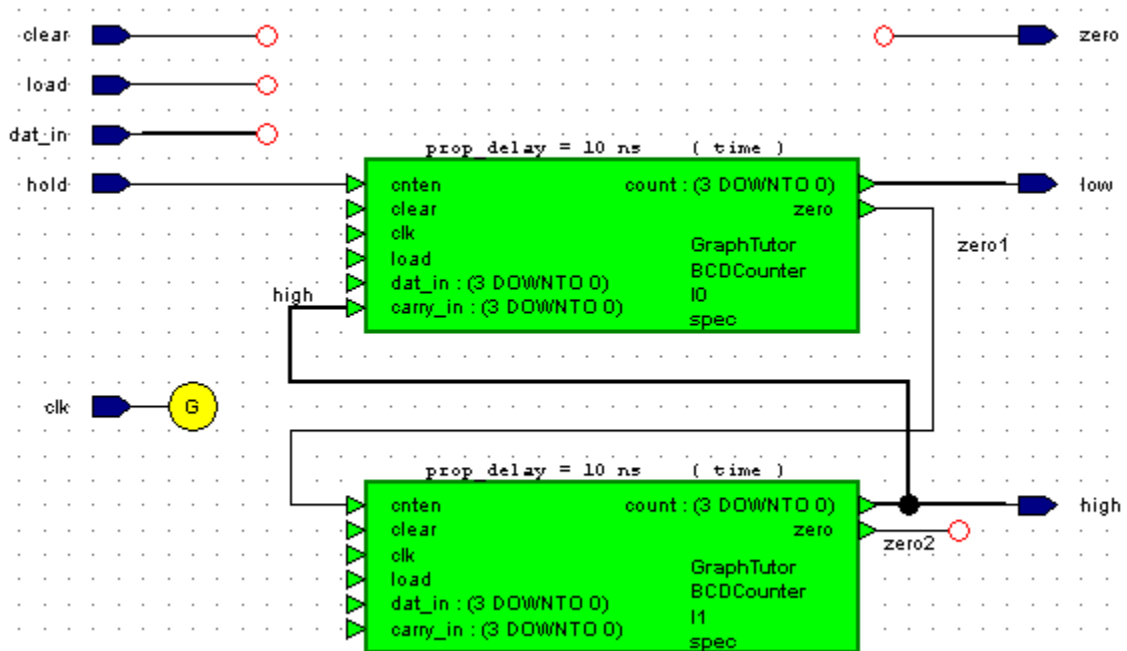


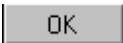
- Drag two instantiations (*I0* and *I1*) of the *BCDCounter* component from the Component Browser placing one below the other on the block diagram but allowing several grid lines for connections between them.
- Move the *hold* signal so that it overlaps the *cnten* port on instance *I0*.



- Similarly, move the *low* signal over the *count* output port on instance *I0* and the *high* signal over the *count* output port on instance *I1*.
- Select both instances and choose **Connect** from the popup menu to make the connections.
- Use the  button to connect a signal between the *zero* output port on *I0* and the *cnten* port on *I1*. This signal is automatically named *zero1* since the output signal *zero* already exists on the diagram.

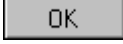
12. Use the  button to connect a bus between a point on the *high* output bus and the *carry_in* port on *I0*.
13. Use the  button to connect a stub signal to the zero output port on *I1*. This signal is automatically named *zero2* since the signals *zero* and *zero1* already exist on the diagram. The block diagram should now look similar to the following picture:



14. Select both instances *I0* and *I1* of the *BCDCounter* and choose **Add Signal Stubs** from the popup menu to display the Add Signal Stubs dialog box.
15. Click the  button to confirm that you want to add signal stubs to all port types. You are warned that the nets *clear*, *clk*, *dat_in* and *load* already exist on the diagram.

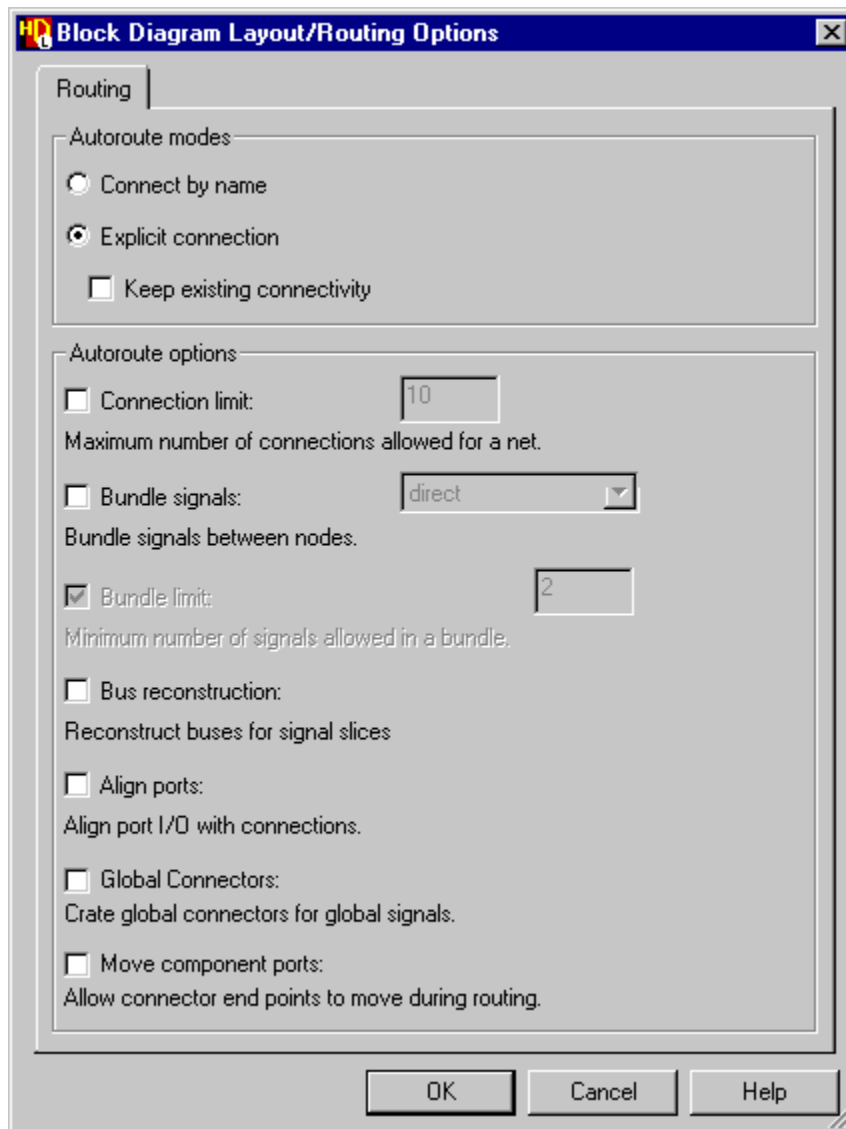


Do not set the **Create unique net names** option. This option could be used when you do not want to automatically connect nets with the same net but is not required in this tutorial.

16. Click the  button to acknowledge the warning. The signal stubs are added when you accept the warning and the matching signals are implicitly connected by name.

It is not necessary to connect the nets on the diagram although making implicit connections can make the diagram clearer. You can automatically connect the matching nets on a diagram by using commands in the **Autoroute** cascade from the **Diagram** or popup menu.

17. Check the options for automatic routing by choosing **Autoroute Options** from the **Autoroute** cascade to display the Block Diagram Layout/Routing Options dialog box.



18. Check that **Explicit connection** is set. Unset **Align ports**, **Global Connectors** and **Move component ports** and confirm the dialog box.

19. Select the *clear*, *load* and *dat_in* nets (or the ports with these names on instances *I0* and *I1*). Choose **Autoconnect** from the **Autoroute** cascade of the popup menu.

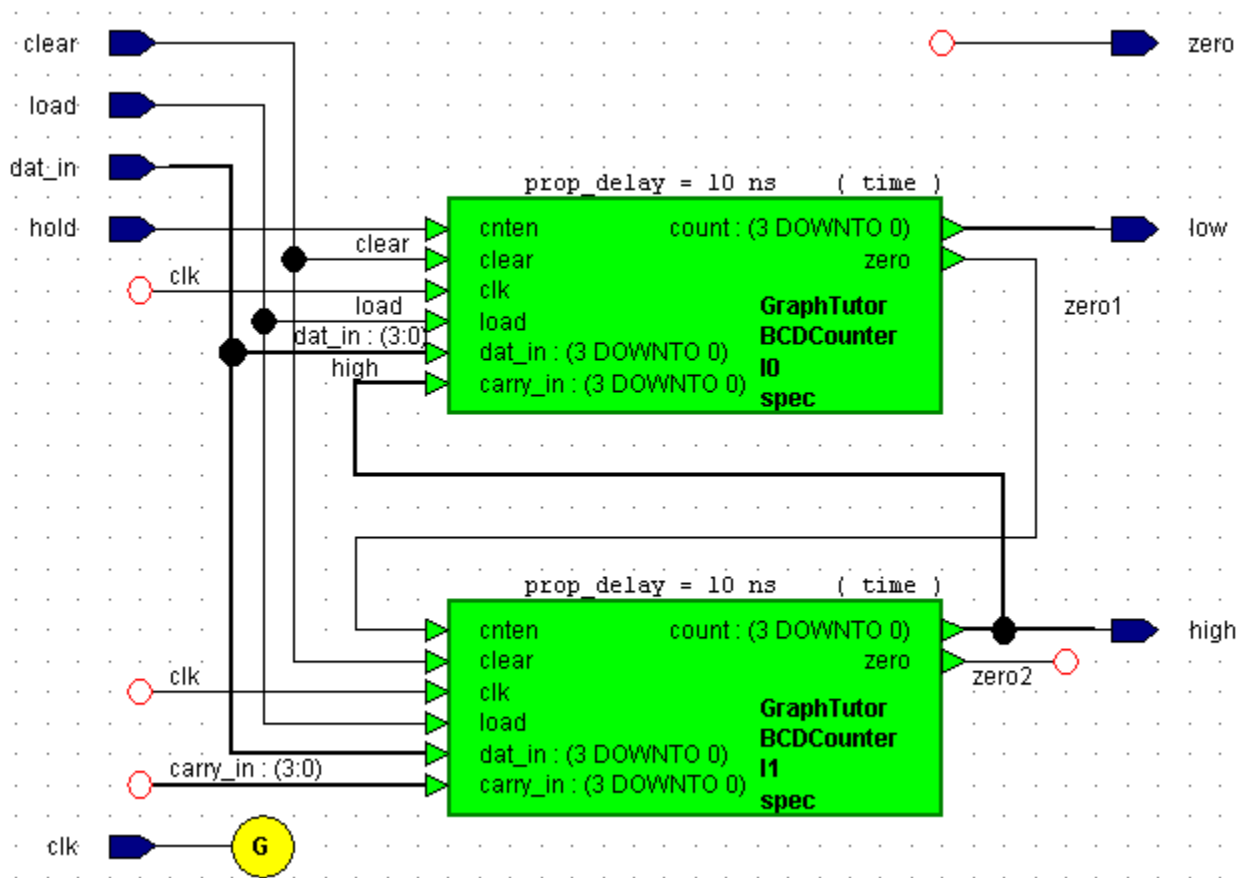
The signals are automatically connected to the matching stub signals on instances *I0* and *I1*.



If nothing is selected on the diagram, autoconnect attempts to re-route all connections on the diagram. Ensure that only the required signals are selected if you do not want to change all the connections on a diagram.


Note that the *clk* port (with its global connector) is connected by name to both *clk* ports on the instances.

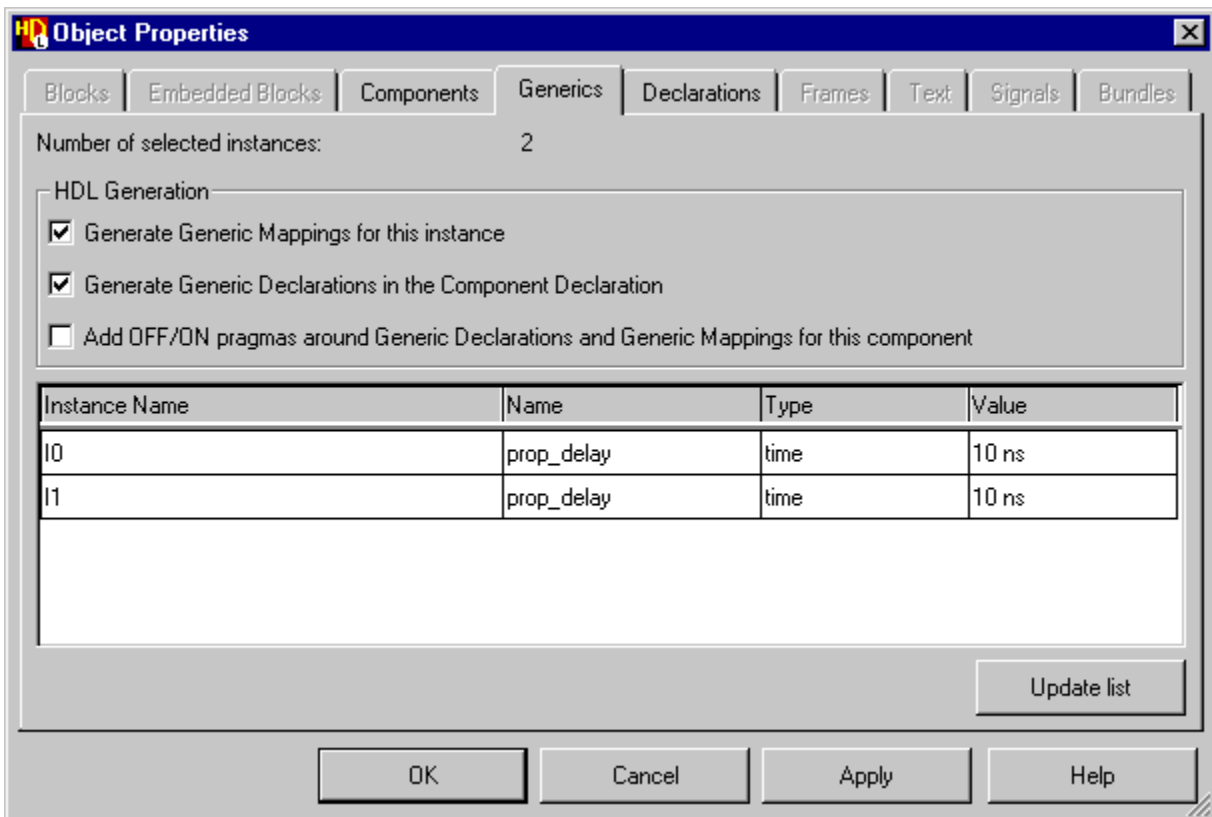
The block diagram should now look similar to the following picture:



Edit the Generic Mapping

When you instantiated the *BCDCounter* components, a VHDL generic mapping for the propagation delay (*prop_delay*) was placed above the component. The generic is declared in the symbol defining the component and can be mapped to different values for each instance of a component. The generic mapping is a text object which can be moved independently by dragging it with the **Left** mouse button. For this tutorial, edit both values should be set to *5 ns*.



- Use the **Shift** key with the **Left** mouse button to select both instances of the *BCDCounter* component. Then use the  button to display the Block Diagram Object Properties dialog box and choose the **Generics** tab.

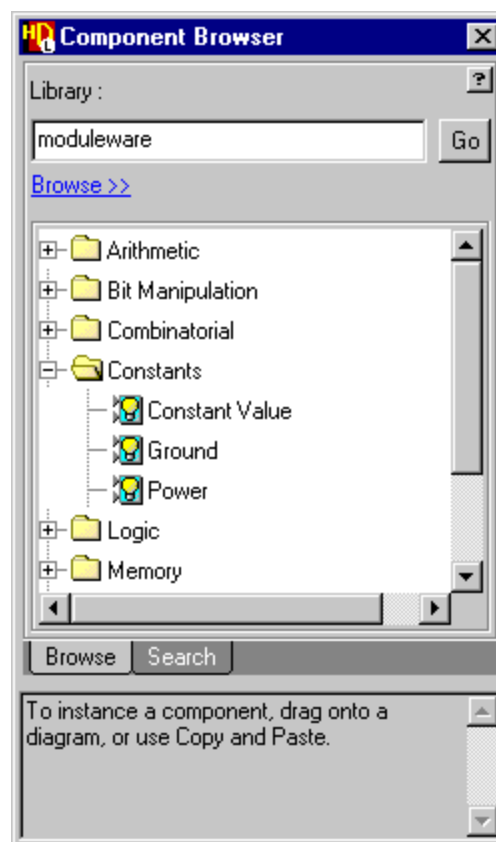



- Select the Value entry for instance *I0* and overwrite it with the new value *5 ns*. Repeat for the value on instance *I1*. Then use the **OK** button to update both values on the block diagram.

Add ModuleWare Components

Although you have routed the *Counter* block diagram, several signals have been left unconnected. These will be connected by using ModuleWare components.

22. Display the Component Browser by using the  button (or by choosing **ModuleWare** from the **Add** menu). The browser displays the contents of the *moduleware* library which is divided into a number of folders.
23. Click on the  icon to expand the folder for the *Constants* category:

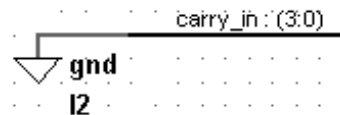


24. Use the  mouse button to drag an instance of the *Ground* component from the Component Browser over the *Counter* block diagram and release the button to place it near the *carry_in* input to instance *I1*.

A *gnd* ModuleWare instance is added with a default instance name (*I1*).

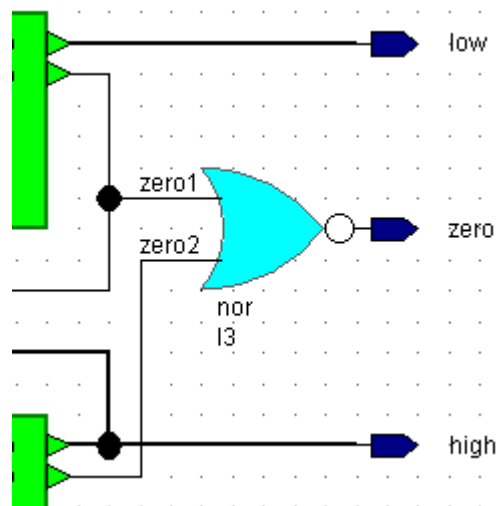
25. Use the mouse to position the stub signal on the *gnd* component over the dangling connector on the *carry_in* net and choose **Connect** from the popup menu.

The *I2* ModuleWare instance should now look similar to the following picture.





Notice that the *carry_in* net is a 4-bit bus. The ModuleWare port width is automatically set to match the width of the connected net.

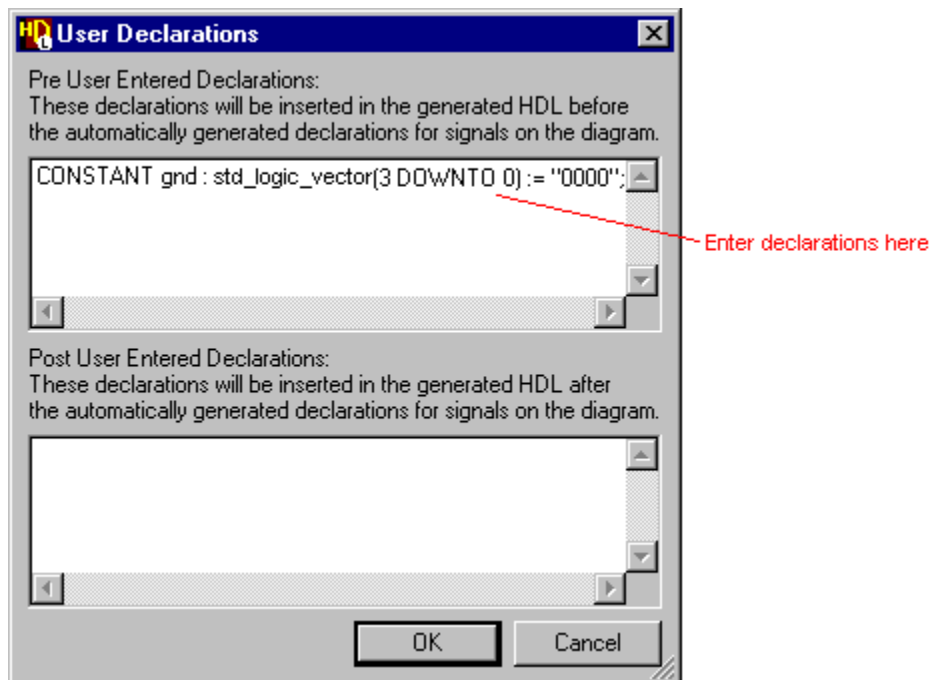
26. Repeat this procedure to add an instance of the *N Input NOR* component from the *Logic* folder of the *moduleware* library near the *zero1* output from instance *I0* on your block diagram.
27. Connect this *nor* instance to the *zero1*, *zero2* and *zero* nets as shown in the following picture:




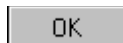
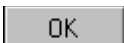
Add a User Declaration

28. Double-click over the Declarations label on the diagram (or use the  button) to display the **Declarations** tab of the block diagram Object Properties dialog box.
29. Click the  button to display a free format entry dialog box. Define the *gnd* signal to be a 4-bit binary zero vector constant by entering the following declaration in the Pre User Entered Declarations section of the dialog box:

```
CONSTANT gnd : std_logic_vector(3 DOWNTO 0) := "0000";
```



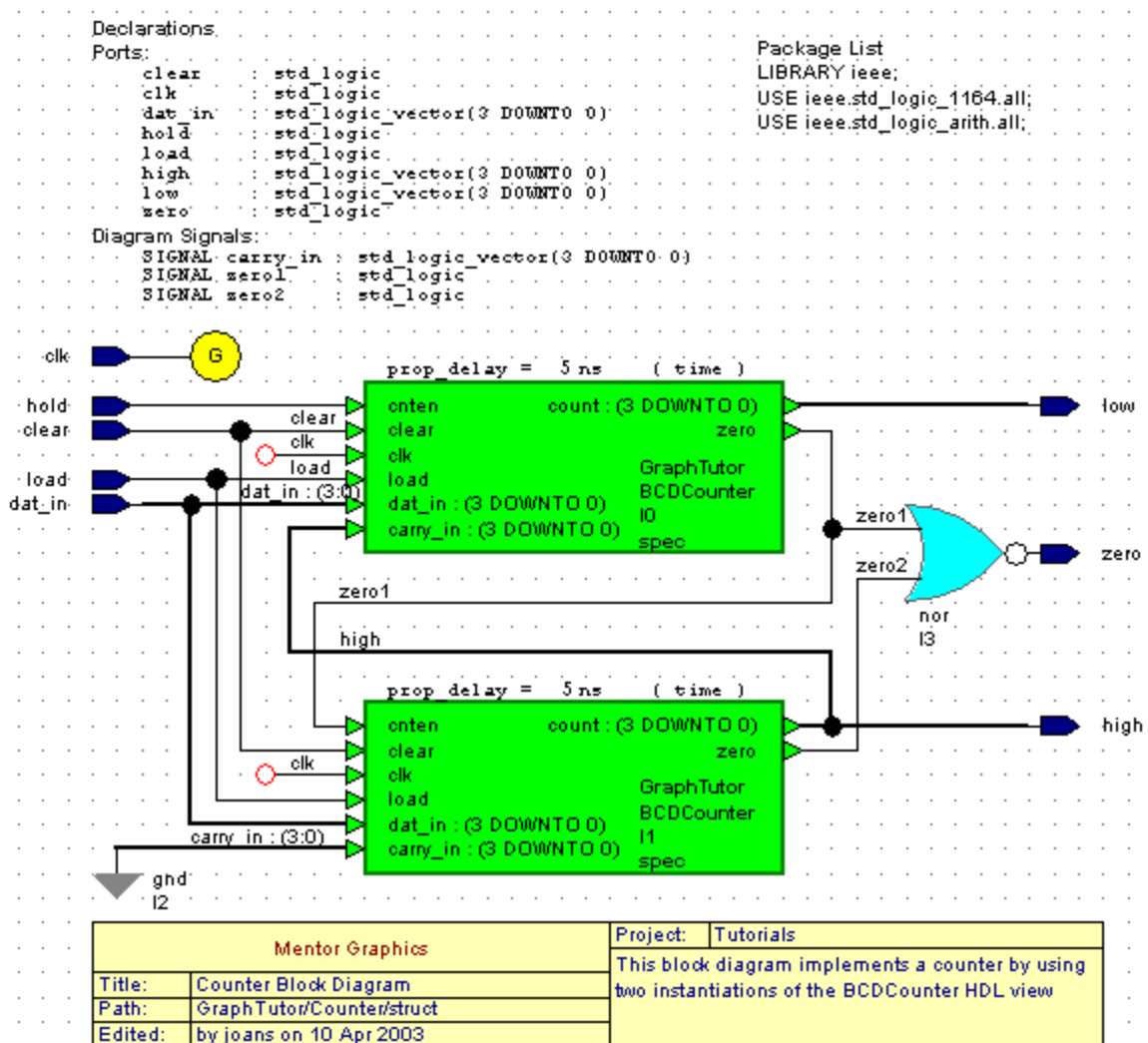
-  The syntax is automatically checked and any errors reported on entry. You can enter any valid VHDL statements which must be terminated by a semicolon (;) character. Line breaks and spaces can be used for clarity and will be preserved on the diagram.


30. Click on the  button to confirm the user declaration and click the  button in the Block Diagram Object Properties dialog box to add the new user declaration on the diagram.

The pre user entered declarations are added to the declarations list on the diagram between the port and signal declarations.

- Complete the block diagram by editing the title block. You may also want to drag objects or groups of objects to re-arrange the diagram within the page boundaries for your default printer.

For example, the following picture shows the *Counter* block diagram re-arranged for portrait orientation.




- Use the  button to save the block diagram.

Create a Truth Table

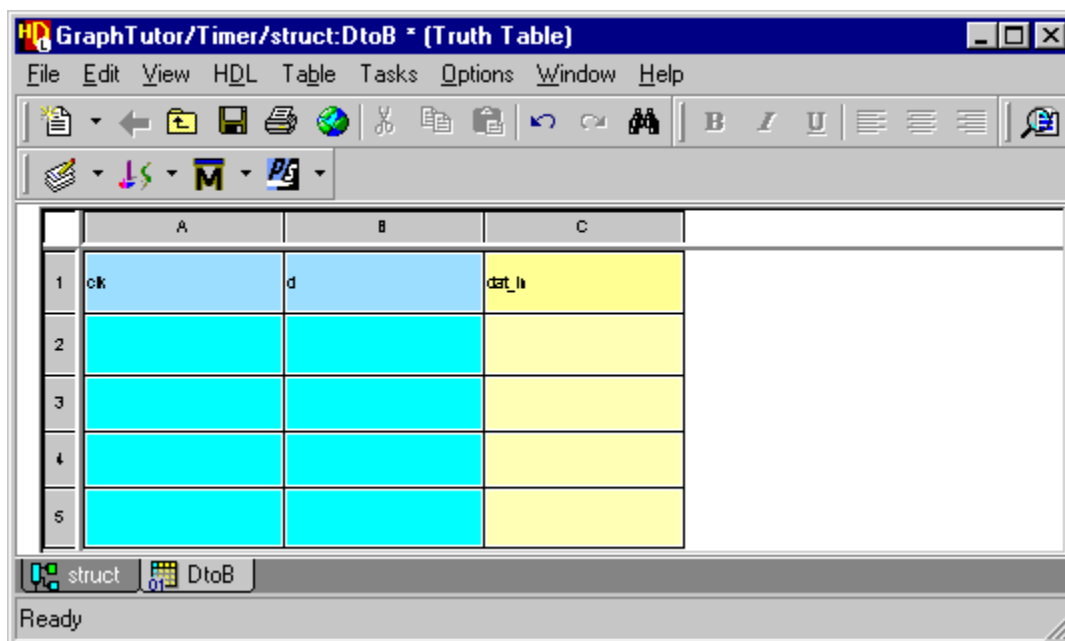
You have now defined a state diagram view for the *Control* block and a block diagram view for the *Counter* block.

The *Timer* design is completed by using a truth table to describe the *DtoB* embedded block.

1. Use the  button (or choose the **Open Up** option from the **File** menu) in the *Counter* block diagram to display its parent diagram (the *Timer* block diagram).
2. Double-click over the *DtoB* embedded block to display the Create Embedded View dialog box.
3. Select **Truth Table** and confirm the dialog box.

A new truth table (*GraphTutor/Timer/struct:DtoB*) embedded view is created and displayed in a tab window.

The table is initialized as a matrix of four rows with two input columns (for the *clk* and *d* inputs) and one output column (for the *dat_in* output).



Edit the Truth Table

- Click the mouse in either of the blue input columns (A or B) and use the **Add Column** option from the popup menu to add another eight input columns (C to J).
- Click the mouse in one of the blue input rows (rows 2, 3, 4 or 5) and use the **Add Row** option to add another seven rows (6 to 12) to the truth table.



You can add multiple rows or columns by selecting more than one cell to add the corresponding number of rows or columns.

- Rename the input columns to represent each bit ($d(9)$ down to $d(0)$) of the d input bus and enter the value '1' in one row for each column. (The clk signal is not used and can be renamed.)









Enter the following values in the dat_in output column:

```

"0000"
"0001"
"0010"
"0011"
"0100"
"0101"
"0110"
"0111"
"1000"
"1001"
"0000"

```

To add text to a cell, click in the cell, enter the text and click in any other cell to confirm the entry. You can edit the text in a cell by double-clicking the cell and overwriting its contents or click again to edit individual text characters.

You can also use the  button to copy and the  button to paste text between cells and re-size the rows or columns by dragging the control sash in the non-scrolling area. You can format text in single or multiple selected cells by using the  (bold),  (italic) or  (underline) buttons and align text using the  (align center),  (align left) or  (align right) buttons.

The completed truth table should look similar to the picture below.

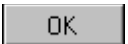
	A	B	C	D	E	F	G	H	I	J	K
1	d	d	d	d	d	d	d	d	d	d	dat_h
2										'1'	'0000'
3									'1'		'0001'
4								'1'			'0010'
5							'1'				'0011'
6						'1'					'0100'
7					'1'						'0101'
8				'1'							'0110'
9			'1'								'0111'
10		'1'									'1000'
11	'1'										'1001'
12											'0000'

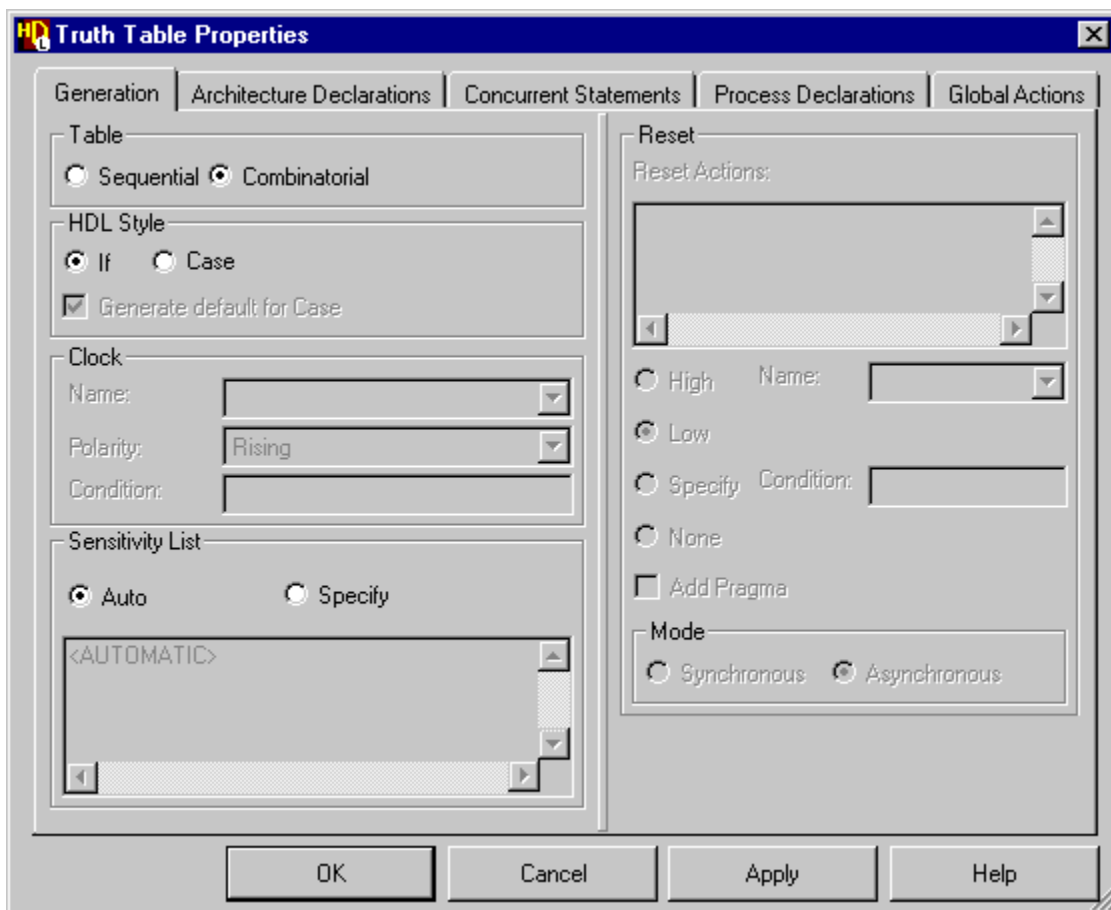


The last (empty) row represents an ELSE condition and assigns the output expression a known value when none of the input expressions are true.

Set Truth Table Properties

7. Choose **Truth Table Properties** from the **Table** menu to display the Truth Table Properties dialog box. Select the **Generation** tab to display the HDL generation characteristics for the truth table.

Notice that the Sensitivity List is set to **Auto**. When this option is set, the sensitivity list is automatically created by HDL generation. Check that the default options for **Combinatorial** and **If** style are set and use the  button to confirm the dialog box.



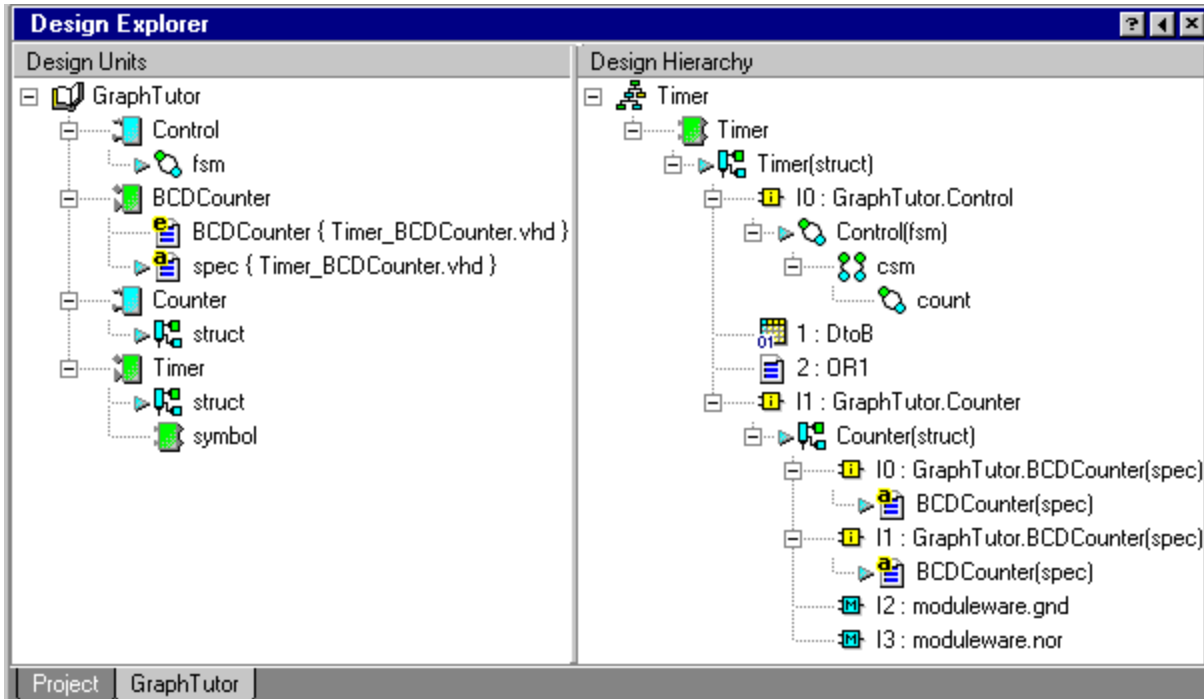
The tabs for Architecture Declarations, Concurrent Statements, Process Declarations and Global Actions can be left empty for this tutorial.

8. Save the truth table. This also saves the *Timer* block diagram since the truth table is an embedded view within the same design unit.

Browse the Timer Design

1. Examine the completed design in the design explorer *Design Units* and *Design Hierarchy* panes using the \oplus icons to expand each design unit.


The fully expanded design should look similar to the following picture:



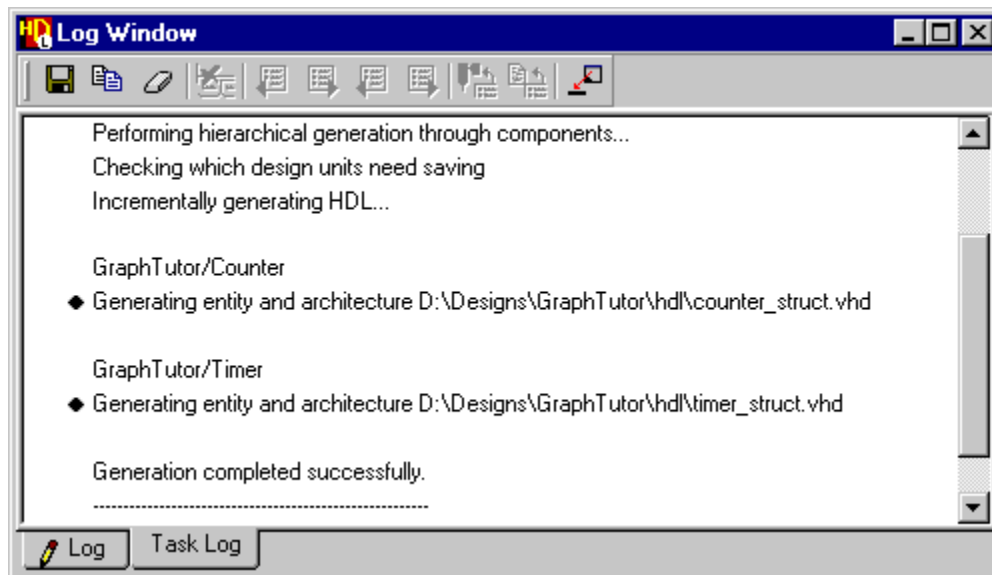
The expanded *Control* block design unit shows that it contains a state machine (*fsm*). The *BCDCounter* component contains a VHDL entity and architecture (*spec*) defined in the *Timer_BCDCounter.vhd* file. The *Counter* block contains a block diagram (*struct*). The *Timer* component contains a block diagram (*struct*) and a symbol describing its external interface.

Notice that you can expand the *Timer* component in the *Design Hierarchy* pane to display the complete hierarchy including the views describing the *BCDCounter* and the *Control* state machine. You can expand the state machine instance to show the child hierarchical state machine (*count*). You can expand the *Counter* instance to show it contains two instantiations of the *BCDCounter* plus the *gnd* and *nor* ModuleWare instances.

Generate HDL for the Hierarchy





2. Select the *Timer* design unit in the *Design Units* or *Design Hierarchy* pane of the design explorer window and choose the  button (or choose **Run Through Components** from the **Generate** cascade of the **Tasks** menu).

The progress of the HDL generation is monitored in a log window which includes any errors and warnings issued during generation.



This example shows a combined file generated for the VHDL entity and VHDL architecture body. Separate files may be generated if you have changed the preferences for VHDL file options.

The *Control* design unit generated earlier in this tutorial is not regenerated unless it has been changed. HDL is generated for all other design units. You are prompted to save any design units which have not been saved.

If there are any errors, you can move to the next error message using the  button or the previous error using the  button. You can display the source graphics corresponding to the error by double-clicking on the error message (or using the  button when the error is selected). Alternatively, you can use the  button when the error is selected to view the generated HDL. If your text editor supports a line number argument, the HDL line corresponding to the error is selected automatically.



The embedded truth table view is generated as part of the *Timer* design unit. If any syntax errors are issued for the truth table, the corresponding cells are highlighted. For example, if HDL generation encounters an expression which is not defined in the diagram interface, the corresponding input cells are highlighted.

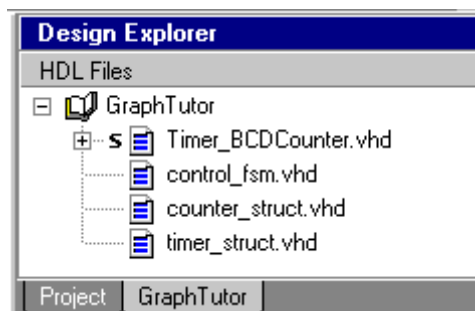
If there are no errors, you can use the toolbar buttons to view the Graphics or HDL corresponding to the “Generating...” message in the HDL Log window.



You can cross-reference to the graphics or HDL from any message which is marked by the ♦ icon in the log window.

Any files in the generated HDL directory can also be opened from the HDL browser.

3. Use the  button (or choose HDL Files from the **Mode** cascade in the **View** menu) to display the *HDL Files* view in the design explorer.
4. Click on the  icon for the *GraphTutor* library and notice that the view is expanded to reveal the source file for the *BCDCounter* and the generated files for the *Control*, *Counter* and *Timer* design units.



You can open any of these files by double-clicking on the generated file name or by choosing **Open** from the popup menu.




You can also view the generated HDL files for the active diagram by using the  button in any editor window or when the diagram view is selected in the design explorer.

Edit the Timer Symbol

1. Use the  button (or choose the **Interface** option from the **Open** cascade of the **File** menu) in the *Timer* block diagram.

A symbol editor window is opened which shows the external interface for the *Timer* design as a default symbol. This symbol is used when the *Timer* design is instantiated as a component in a higher level design.

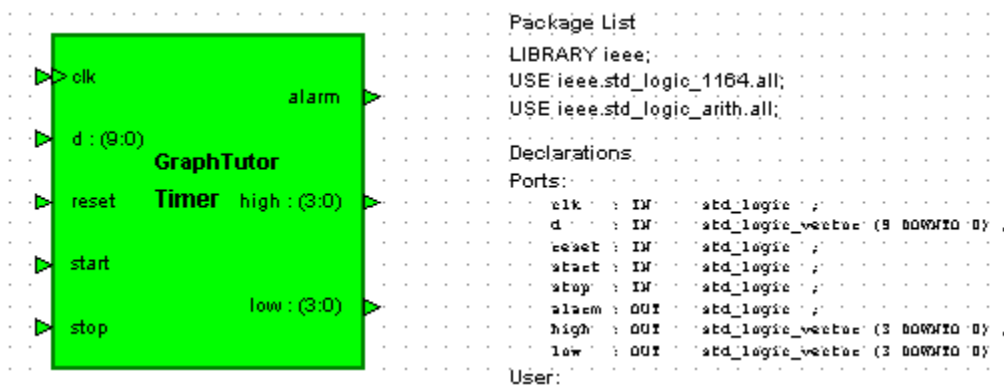
Ports representing the interface connections are shown in default locations but you can select and drag the ports to any location around the body of the symbol. You may also want to resize the symbol body.

-  You can redistribute the ports evenly by selecting the symbol body and choosing **Equidistant Ports** from the popup menu.

The full port declarations are shown as a list in the symbol editor and are not shown on the individual ports. Refer to “Changing the Visibility of Symbol Port Properties” in the *HDL Designer Series User Manual* for information about setting preferences to control whether type and bounds information is displayed or the properties visible for an individual port.

2. Select the *clk* port and choose **On** from the **Clock** cascade to apply an edge triggered clock indicator to the port.

The edited symbol should look similar to the picture below:



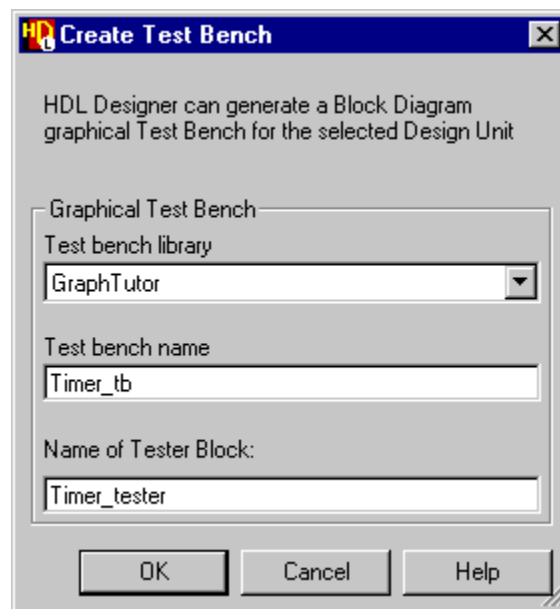
3. Close and save the symbol.

Create a Test Bench


1. Select the *Timer* design unit in the *Design Units* view of the design explorer and choose **Create Test Bench** from the **New** cascade in the **File** menu to display the Create Test Bench dialog box.

Notice that the dialog box defaults to the *GraphTutor* library, with the test bench name derived by adding *_tb* to the *Timer* design unit and the default tester block by adding *_tester*.

2. Confirm the dialog box by clicking the button.



Notice that a *Timer_tb* design unit is added to the *Design Units* displayed in the design explorer window.

3. Click on the  icon for the *Timer_tb* design unit in the design explorer to expand the view and reveal that a symbol and block diagram view (with the default name *struct*) have been created.
4. Double click on the *struct.bd* view to open the block diagram.

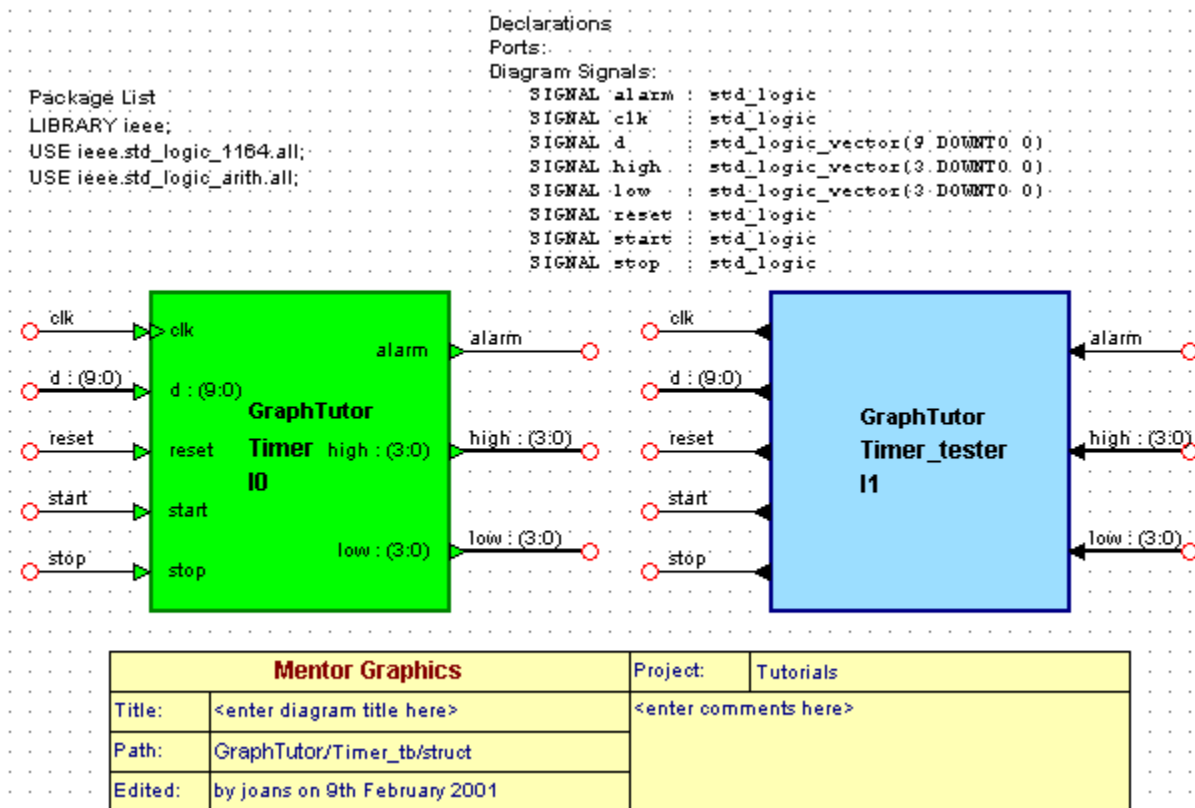
Notice that the *Timer* design has been instantiated as a component with stub signals connected to the input and output ports. The adjacent *Timer_tester* block has corresponding output and input ports which are implicitly connected by name to the stub signals on the component.



The port locations are identical on the *Timer* component and *Timer_tester* block but their directions are reversed. Hence the output ports are on the left and input ports on the right of the block.

All the signals and buses are declared as diagram signals since a test bench typically has no external ports.

The test bench block diagram should look similar to the following picture.




It is not necessary to explicitly connect signals and buses between each port on the component and the tester block as these are implicitly connected by name.

Import the Tester Design Unit

The *Timer_tester* block should provide test stimulus to the *Timer* design unit and monitor its output.



For this tutorial, the *Timer_tester* design unit can be imported from the *examples* installation subdirectory using a similar procedure to that used to “[Import the BCDCounter Design Unit](#)” on page 1-44.

The *Timer_tester* design unit can optionally be created as a flow chart. Refer to the alternative procedures for [Creating a VHDL Flow Chart](#) which are provided in [Chapter 3](#).

1. Use the  button in the design manager window (or choose **HDL Import** from the **HDL** menu) to display the HDL Import wizard.
2. Choose **Specify HDL files** in the first page and select the *Timer_tester.vhd* file in the Specify HDL Source Files page of the wizard.



The previous HDL import directory and files are saved as preferences. You should only need to remove the *Timer_BCDCounter.vhd* file and add the *Timer_tester.vhd* file.

3. Use the  button to page through the HDL Import wizard and use the  button on the last page to import the *Timer_tester* design unit as a HDL text view.

The following completion message should be displayed in the HDL log window:


```
** Importing 'D:\Designs\GraphTutor\hdl\Timer_tester.vhd'
HDL Import complete
```

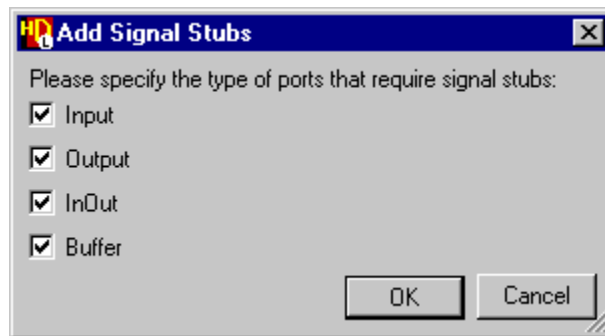
```
-----
```

```
1 file imported to 1 library
```

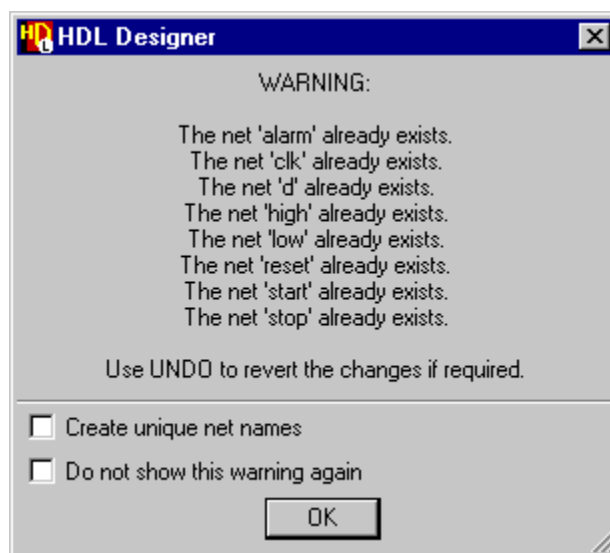
```
-----
```

Instantiate the Imported Tester

4. Re-open the *Timer_tb* block diagram if it is not still open. Select the *Timer_tester* block including all the connected signal stubs and delete them using the **Del** key.
5. Use the  key to display the Component Browser. Select the *GraphTutor* library and instantiate the imported *Timer_tester* component in the block diagram.
6. Select the *Timer_tester* component and choose **Add Signal Stubs** from the popup menu to display the Add Signal Stubs dialog box:



7. Click the **OK** button to confirm that you want to add signal stubs to all port types. You are warned that the nets *alarm*, *clk*, *d*, *high*, *low*, *reset*, *start_in* and *stop* already exist on the diagram.



8. Click the button to acknowledge the warning. The signal stubs are added when you accept the warning and the matching signals are implicitly connected by name.
9. Complete the test bench by editing the title and comments in the title block and using the button to add a panel around the components. This panel can be used to set the diagram view when you simulate your test bench later in this tutorial.

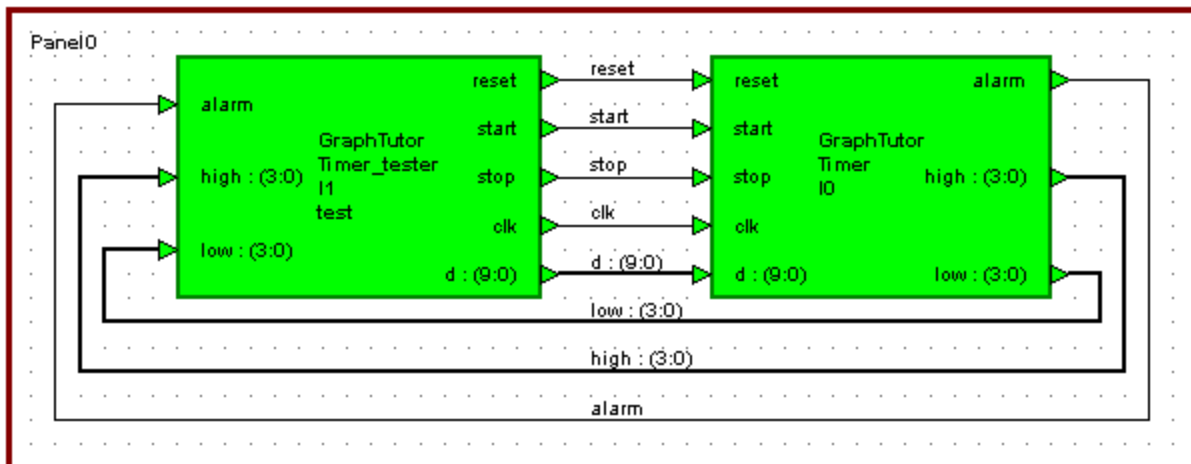
The matching nets on the two components are implicitly connected by name. However, you may like to connect them explicitly by choosing **Autoconnect** from the **Autoroute** cascade of the **Diagram** menu.



Using this command with nothing selected should automatically route all nets on the diagram. If any nets are selected, then only the matching nets in the selection are routed.


It may also be necessary to move some of the component ports in order to avoid cross-overs.

The following example shows a test bench which has been explicitly connected:

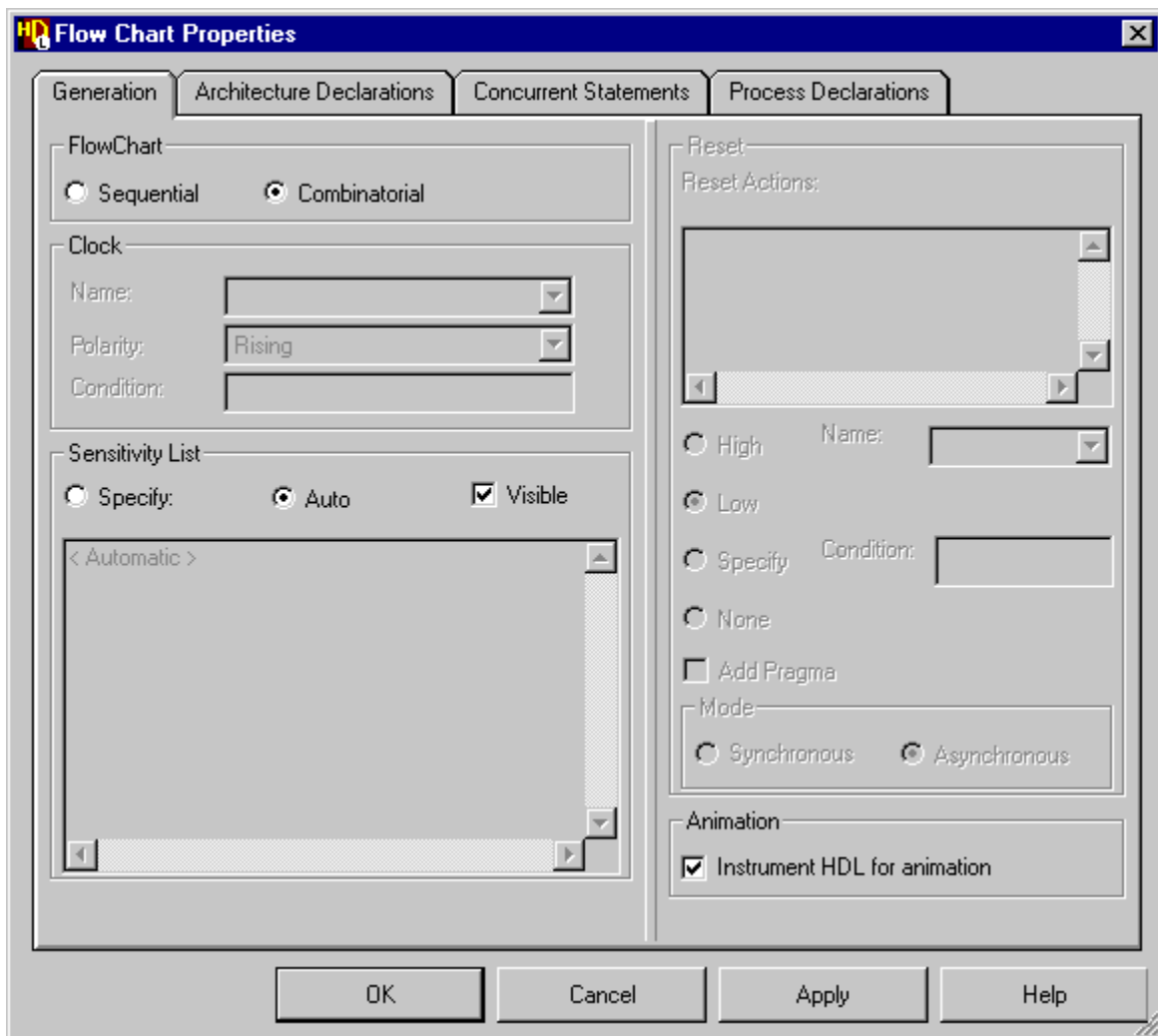


Generate HDL for the Test Bench


10. If the *Timer_tester* component is described by a flow chart, double-click on its instance in the test bench to open the view and choose **Flow Chart Properties** from the **Diagram** menu to display the Flow Chart Properties dialog box.



 If you have recovered the *Timer_tester* component as a HDL text view, all properties for the view are defined in the HDL text and do not need to be set using a dialog box.

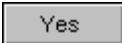
11. Choose the **Generation** tab and check that the **Combinatorial** option is set.



12. If a ModelSim or NC-Sim simulator is available, set the **Instrument HDL for animation** option. This option adds additional code to the generated HDL which is used for flow chart animation later in this tutorial.

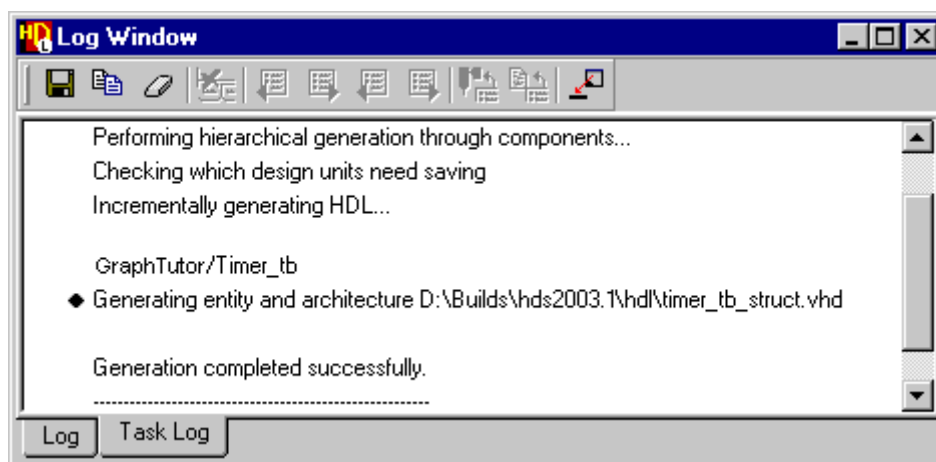
 The additional animation code is surrounded by translation control pragmas which ensure that it is ignored by downstream synthesis tools.

13. Set the **Auto** option for the Sensitivity List and use the  button to confirm the dialog box.
14. Select the *Timer_tb* design unit in the design explorer and choose the  button (or choose **Run Through Components** from the **Generate** cascade of the **Tasks** menu).


All views to be generated must have been saved and if you have not saved the test bench you are prompted whether to continue. Confirm the dialog box by clicking the  button.

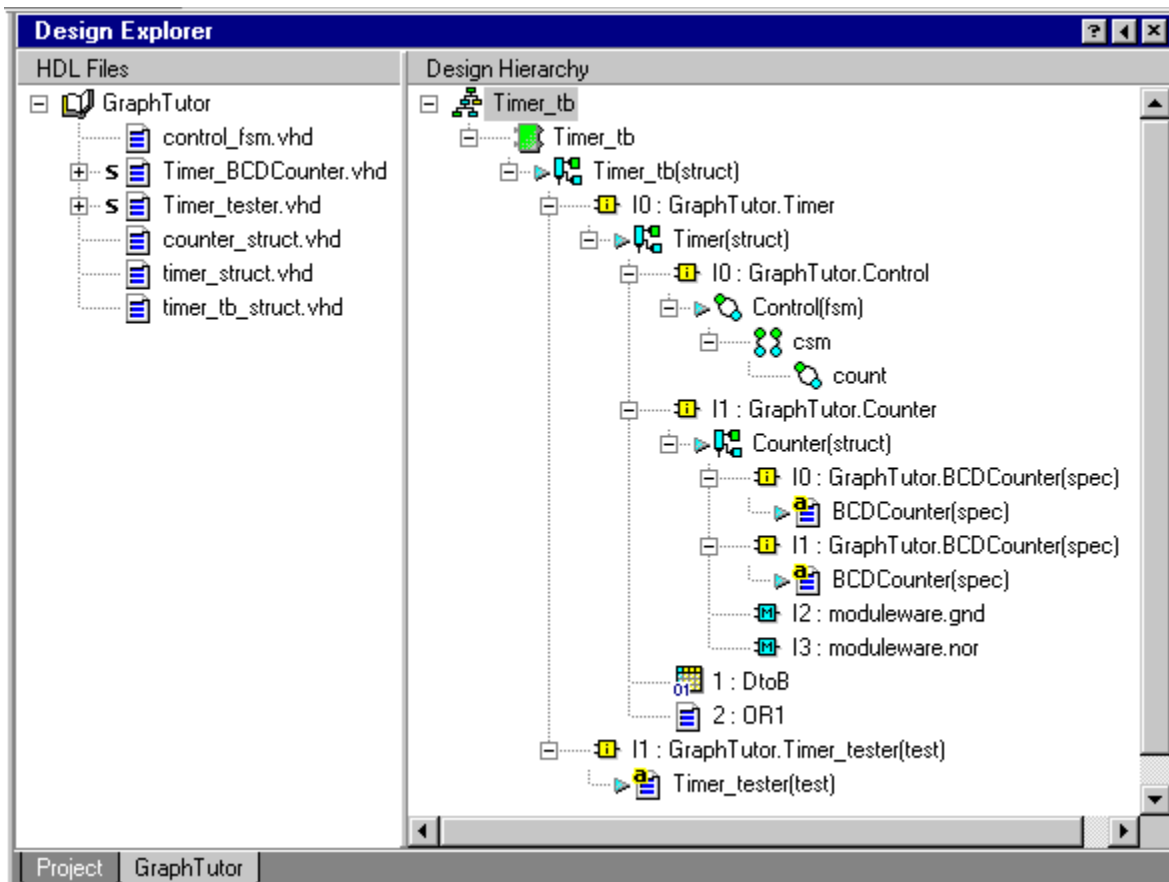
VHDL is generated for the test bench and tester (if it is described by a flow chart) views but not the *Timer* design hierarchy since these views were generated earlier in this tutorial and do not need to be regenerated unless they have been modified.

The progress of the HDL generation is monitored in a log window which includes any errors and warnings issued during generation. If there are any errors, you can display the corresponding graphics by double-clicking on the error message as described in the section [“Generate HDL for the Hierarchy” on page 1-62](#).



Browse the Completed Design

1. Select the *Timer_tb* design unit and choose *Show Hierarchy* from the popup menu to display the *Design Hierarchy* pane of the design explorer.
2. Select the *Timer_tb* design unit in the *Design Hierarchy* pane and choose **Expand All** from the popup menu or use the \oplus icons to expand the full hierarchy which also includes the hierarchy of the *Timer* design.
3. Use the  button to view the *HDL Files* pane which should now show the *Timer_BCDCounter.vhd* and *Timer_tester.vhd* source files plus the generated files for *control_fsm.vhd*, *counter_struct.vhd*, *timer_struct.vhd* and *timer_tb_struct.vhd*.



You can use the Logical Objects pane to explore the design in detail. Refer to the the [HDL Designer Series User Manual](#) for more information.

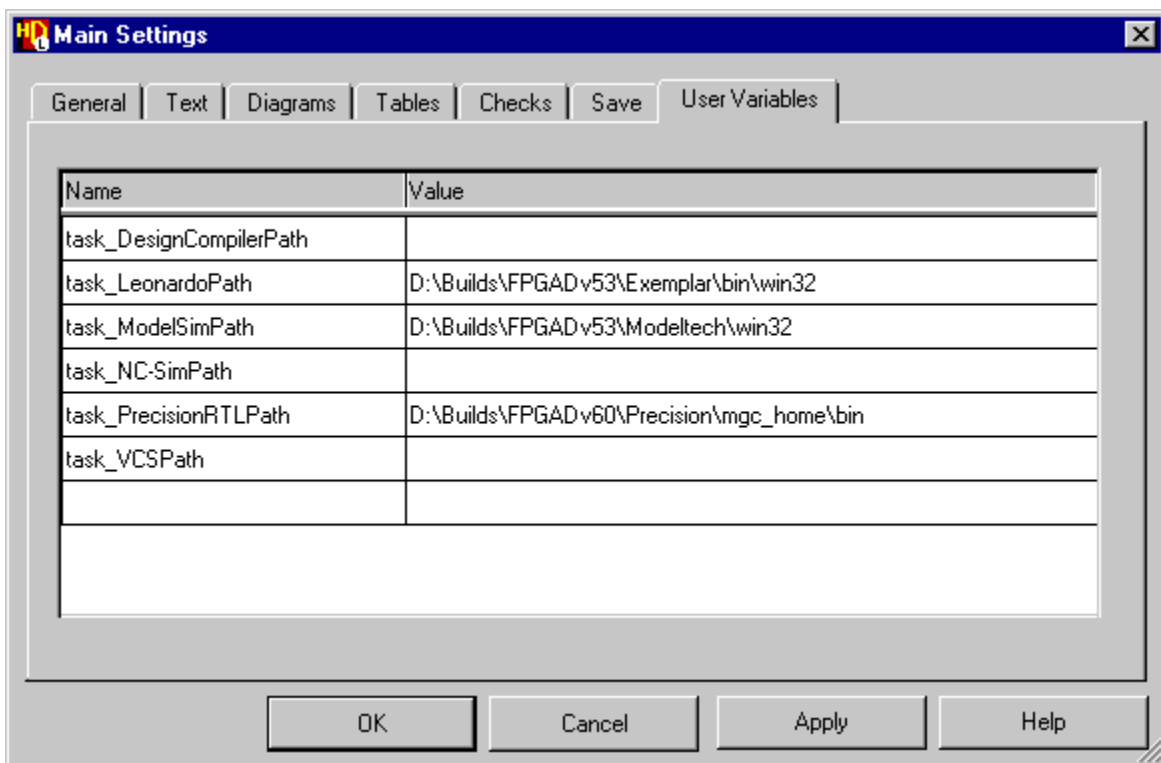
Setup the Downstream Tools

For this tutorial, it is assumed that ModelSim simulator and LeonardoSpectrum synthesis tools are available. You can alternatively prepare data for use with other downstream tools.



If you have installed the HDL Designer Series tool as part of FPGA Advantage, the ModelSim simulator and LeonardoSpectrum synthesis tools will already have been set up automatically.


1. Choose **Main** from the **Options** menu to display the Main Settings dialog box and choose the **User Variables** tab:



User variables are provided for the Design Compiler, LeonardoSpectrum, ModelSim, NC-Sim, Precision Synthesis and VCS (or VCSi) tools.

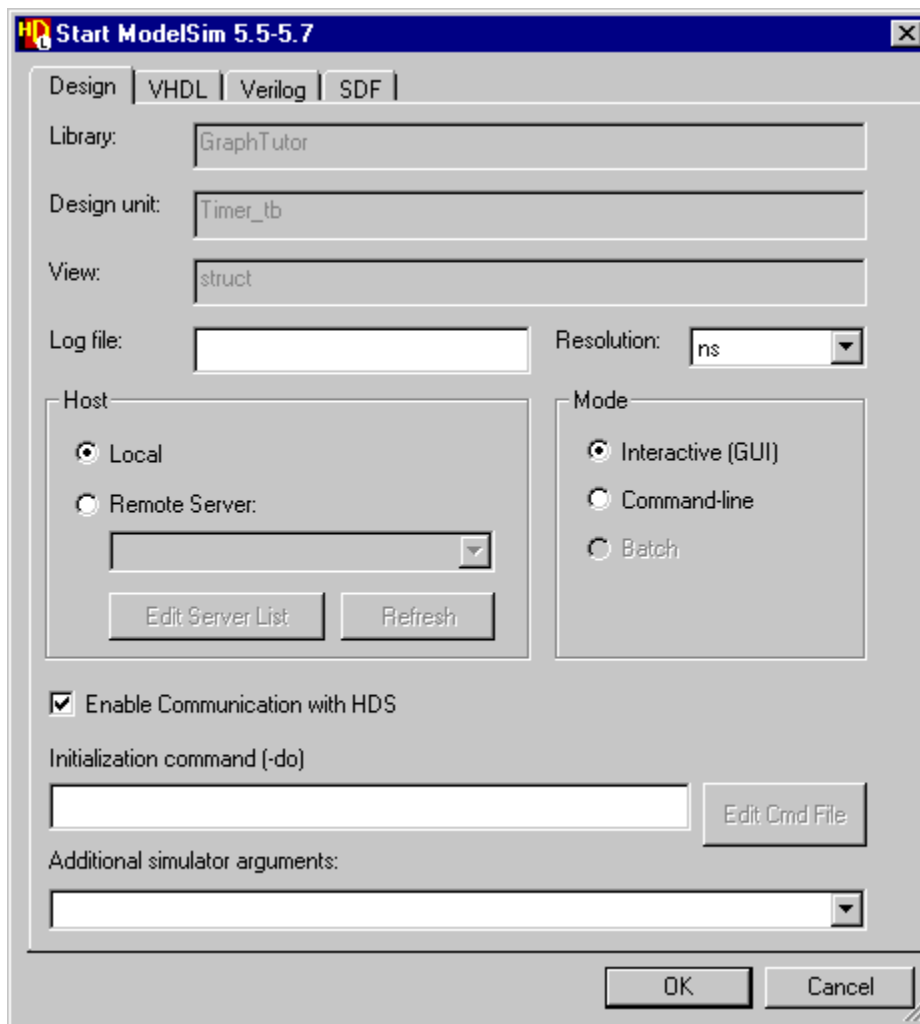
2. Enter the pathname to the executables for any of the downstream tools which are available on your system and confirm the dialog box.

Run the ModelSim Flow

1. Select the *Timer_tb* design unit in the design explorer and use the  button to run the ModelSim flow through components.

HDL is regenerated for any modified graphical views and the entire design is compiled. The progress of the compilation is shown in the HDL Log Window. If there are any errors, you can display the corresponding source diagram by double-clicking on the error message as described in the section [“Generate HDL for the Hierarchy”](#) on page 1-62.

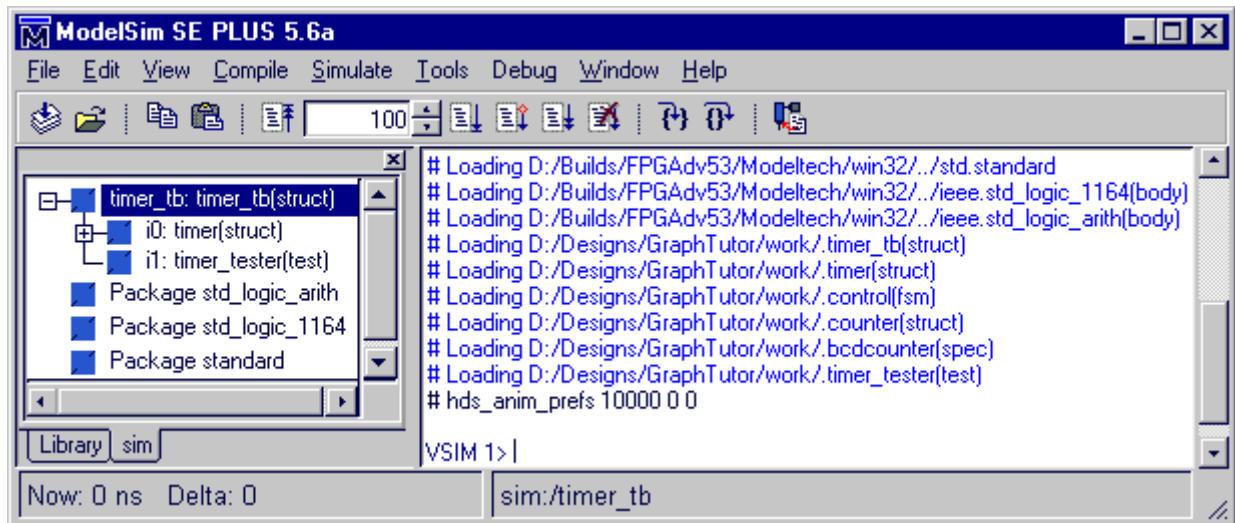
If there are no errors, the Start ModelSim dialog box is displayed:



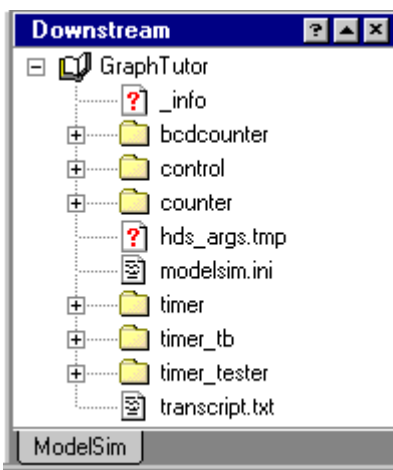
2. Accept the default resolution and check that the **Interactive (GUI)** and **Enable Communication with HDS** options are selected. The other entry fields can be left in their default state.
3. Use the button to confirm the Invoke Settings dialog box.

The simulator is invoked and issues a number of loading messages ending with:

```
# hds_anim_prefs 10000 0 0.
```



Note that library mapping is automatically created for your downstream data and the compiled objects can be displayed in the downstream browser by selecting **Downstream** from the **SubWindows** cascade of the design explorer **View** menu and choosing the ModelSim tab:



Setup the Simulator Windows



An additional Simulation toolbar is available in the block diagram, flow chart and state diagram when a simulator is invoked to support cross-probing between the simulator and source design objects in the HDL Designer Series tool. This toolbar is normally displayed automatically at the bottom of the diagram window but can be undocked, moved, docked or hidden in the same way as the other toolbars.

For example, the following picture shows a toolbar that has been undocked from the editor window and re-arranged as two rows of tool buttons:



4. Display the *Timer_tb* block diagram and choose **View Panel** from the **View** menu to display the panel you added earlier in this tutorial.




i If you added more than one panel, a dialog box is displayed for you to choose the panel to display.

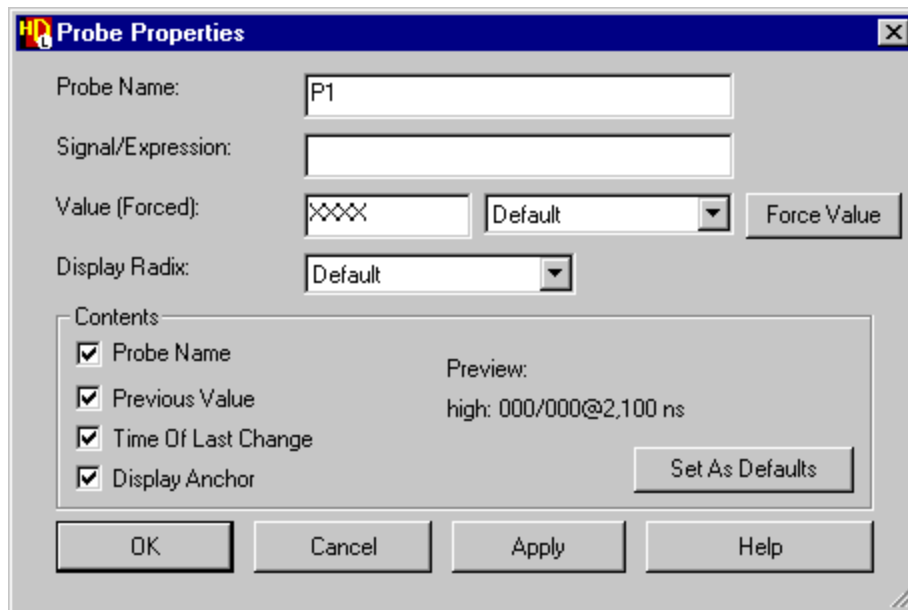
5. Select the signals *alarm*, *clk*, *d*, *reset*, *start*, *stop*, *high* and *low*. Use the  button from the Simulation toolbar or choose **Add Wave** from the **Display** cascade in the **Simulation** menu and notice that the simulator Wave window is automatically opened with these signals loaded.
6. Use the  button from the Simulation toolbar or choose **Add List** from the **Display** cascade in the **Simulation** menu to add the selected signals to the simulator list window.


i You can optionally add signals to the simulation log without displaying them in the Wave or List window by choosing **Add Log** from the **Display** cascade of the **Simulation** menu.

You can also open any other simulator window by choosing from the **View** cascade of the **Simulation** menu. For example, you may wish to use this menu to open the ModelSim **Source** and **Structure** windows.

Add Simulation Probes

- Set simulation probes on the *low* and *high* signals by using the  button or by choosing **Add** from the **Probes** cascade of the **Simulation** menu. The probes are shown by  icons with text displaying the current values.
- Select the probe on the *high* signal and use the  button in the Simulation toolbar to display the Probe Properties dialog box.



- Change the probe name from the default net name to *P1* and set the Contents options for **Probe Name**, **Previous Value** and **Time of Last Change**. Use the  to update the probe display on the block diagram.
- Repeat this procedure for the probe on the *low* signal renaming this probe as *P2*. Both probes should now display the probe names, current and previous values (initially unknown) and the current simulation time (initially 0 nanoseconds):



The probes are connected to the signal nets by an anchor and can optionally be moved to improved diagram clarity.

Enable Animation

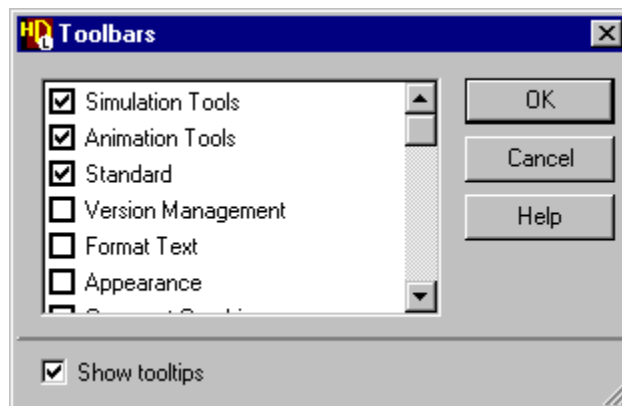
- Open the *Control* state diagram and its child hierarchical state diagram. Use the **View Panel** command in each tab to adjust the window view to the panel areas you defined earlier in this tutorial.

i If you have created a flow chart view for the *Timer_tester* view, it can be animated in a similar way to that described below for the state diagrams.

Notice that an additional Animation toolbar is available in the state diagram (and flow chart) windows when the simulator is invoked. This toolbar is normally docked at the bottom of the diagram window next to the Simulation toolbar but can be moved independently.




The editing toolbars are not usually required during simulation and animation. You can hide or show toolbars using the Toolbars dialog box which is displayed by choosing **Settings** from the **Toolbars** cascade of the **View** menu.




- For example, use the dialog box to hide the Format Text, Appearance, Comment Graphics and Arrange Object toolbars.


i Individual toolbars can also be hidden by unsetting the corresponding options in the **Toolbars** cascade of the **View** menu.

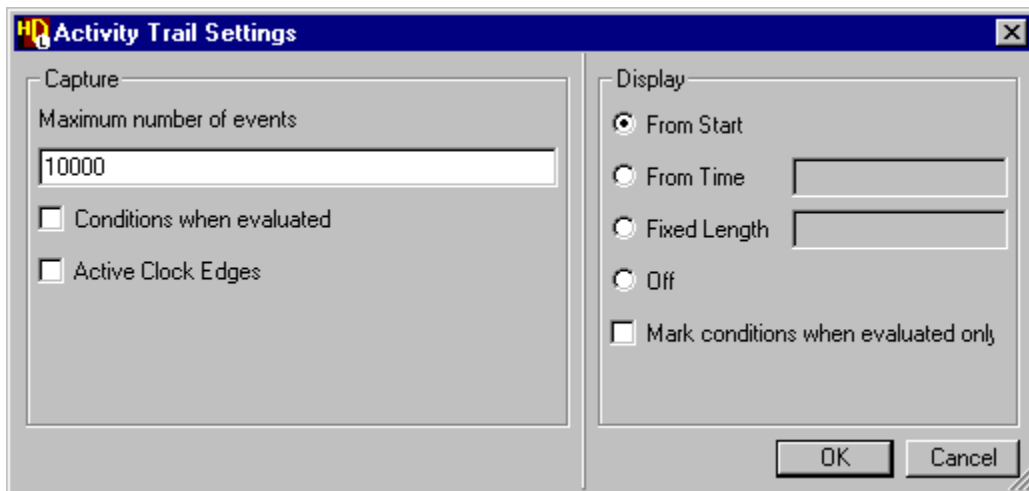
13. Use the  buttons from the Animation toolbars (or choose **Data Capture** from the **Animation** menus) in each state diagram or flow chart you want to animate. (It is not necessary to repeat this command for each hierarchical or concurrent view since these are considered part of the same diagram.)

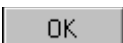
Notice that all objects (except the start state, exit and entry points in the state diagram and the start point in the flow chart) are redrawn with white fill. The start state and start point are drawn with red fill to indicate that they represent the current animation state.



This animation view is automatically displayed when you set data capture but can be toggled at any time by using the  button (or by toggling the **Show Animation** option in the **Animation** menu). You can also set data capture for all instances in the current simulation hierarchy by choosing **Global Capture On** from the **Animation** menu.

14. Use the  button from the Animation toolbar (or choose **Activity Trails** from the **Animation** menu) in the flow chart or state diagram window and choose the **From Start** option in the Activity Trail Settings dialog box.



15. Use the  button to confirm the Activity Trail Settings dialog box.




The activity trail is applied to all flow chart or state diagram windows in the current simulation.



16. Use the  button or choose **Add State Variable to Wave Window** from the **Display** cascade of the **Simulation** menu in the state diagram.

This command monitors the state machine variable by a signal corresponding to the current state of the state machine.



17. Use the  button or the **Add State Variable to List Window** command to add the same signal in the List window.


Simulate the Design


18. Run the simulator for the default timestep (100 nano-seconds) by using the  button or by choosing **For Time** from the **Run** cascade of the **Simulation** menu in the block diagram, flow chart or state diagram

 The default timestep can be changed by using the  button to select from a list of alternative timesteps or choose a user specified timestep.


Notice that the *Initialization* action box is highlighted red in the animated flow chart for the *Timer_tester* and the signal values are updated in the simulation probes on the block diagram and in the simulator Wave and List windows.

19. Run the simulator for another default timestep and notice that the P1 probe is shown by a yellow icon because its value is unchanged since the last simulator timestep.
20. Select the *stop* signal in the *Timer_tb* block diagram and set a breakpoint by using the  button or choosing **Add** from the **Breakpoints** cascade of the **Simulation** menu. The breakpoint is shown by a  icon.

 You can optionally add a simulation probe to the *stop* signal using the procedure described in [“Add Simulation Probes” on page 1-78](#).


21. Run the simulator until there are no more events scheduled by using the  button from the Simulation toolbar or by choosing **Forever** from the **Run** cascade of the **Simulation** menu.

The simulation should run until the breakpoint is encountered when the value of the *stop* signal value (shown in the Wave and List windows) changes to '1'. Notice that the *Store* action box is highlighted in red as the current step in the animated flow chart and the *counting* state is entered in the child animated state machine for the *count* state.



22. Use the  button or choose **Continue** from the **Run** cascade of the **Simulation** menu to continue the simulation until the *stop* signal changes back to zero.

Notice that the *suspended* state is entered in the animated state machine and the following note is issued in the main simulation window:

```
**Note: Count suspended correctly.
```


23. Run the simulator to the next change of the *stop* signal by using the  button and notice the message:


```
**Note: Alarm asserted correctly
```



24. Use the  button in the test bench block diagram to delete the breakpoint on the *stop* signal and complete the simulation by using the  button. The simulation should run to completion and issue the message:

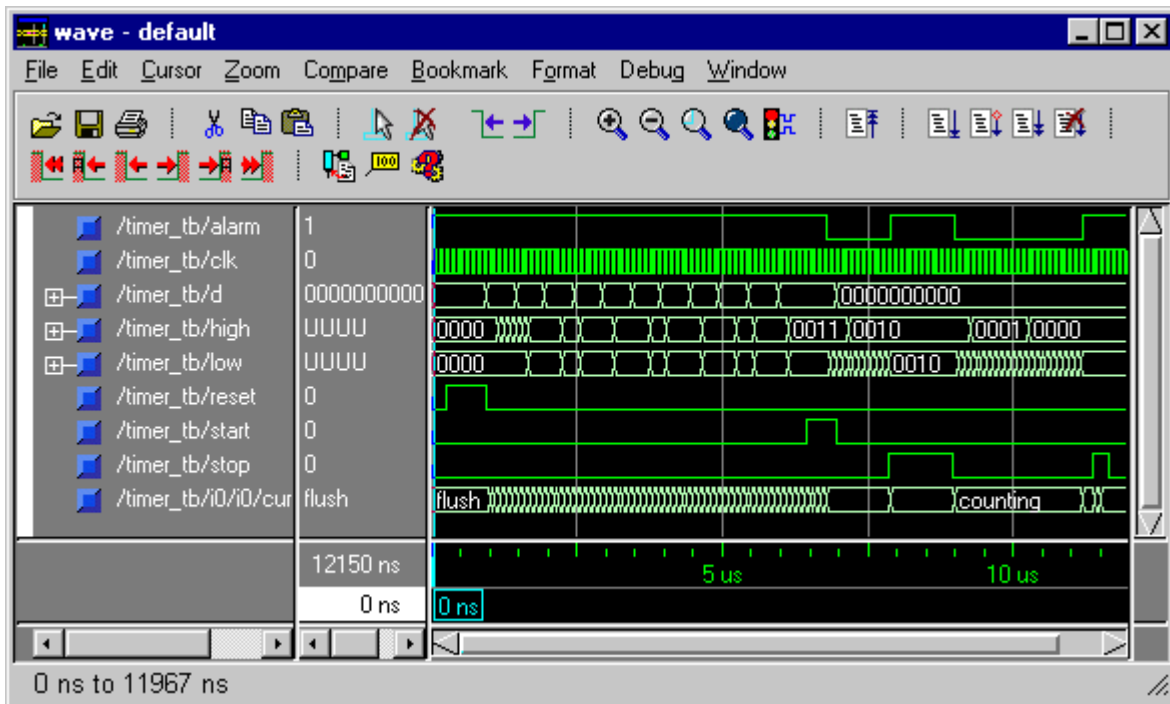
```
**Failure: Timer test completed
```


Examine the results displayed in the Wave window and ensure that the simulation has performed correctly by comparison with the example waveforms shown on the next page.

If any VHDL errors are discovered, correct your design and use the  button to regenerate and recompile the modified diagram. If the diagram is successfully generated and compiled, the flow button automatically restarts the simulation for the current simulation hierarchy.

Alternatively, you can use the  button or choose **Restart Simulator** from the Simulation menu to repeat the simulation.


-  You can use the  button or choose **Highlight Object** from the **Display** cascade of the simulation menu to highlight all occurrences in the simulation windows of a signal corresponding to any selected graphical object.













You can display the full simulation waveform by using the  button or choosing **Zoom Full** from the Wave window **Zoom** menu.

Review the Animation

Notice how the animation activity trail in the animated diagrams is highlighted in blue and the exit break point for `Timer_test_completed` is indexed on the flow chart VHDL architecture displayed in the simulator Source window.

25. If you have enabled animation for the `Timer_tester` flow chart, use the  button or choose **Link Diagrams** from the **Animation** menu to link the animated diagrams. When the diagrams are linked, the animation review commands are applied to all the animated flow charts and state diagrams in the simulation hierarchy.

You can review the animation by using the , , ,  or  buttons (or by choosing **Goto Next**, **Goto Previous**, **Goto Time**, **Goto Start** or **Goto Latest** from the **Animation** menu).


The  and  buttons step through each object in the flow chart. However in the state diagram, you can set options in the **Animation** menu (or from a pulldown menu on the toolbar button) to move by change of state , simulation event  or change of clock edge . The toolbar button icon changes to indicate the current mode of movement.


You may want to change the activity trail to display the trail from a specified time or specify a fixed length. The current simulation step (or current state and last transition taken in an animated state diagram) is always shown in red and the previous step (or previous state and transition) in yellow. All other visited objects included in the activity trail are shown in blue. Remember that you can open down the hierarchical action box or hierarchical state to view the animation in the child diagrams.

26. Use the toolbar buttons to move forwards and backward through the animation.
27. Compare the animation with the results displayed in the Waveform window and ensure that the simulation has performed correctly by comparison with the example waveforms on the previous page.

Debugging From ModelSim



If you are using ModelSim, additional toolbar and menu commands are provided for cross-probing to the graphical views. These include the following:


A  button is available in the Animation toolbar and a **Cause** option from the **Animation** menu. These commands can be used to move the ModelSim Wave and List windows to the current animation time.


A corresponding  button and **Cause** option (in the **Debug** menu) are available in the ModelSim Wave window. These commands can be used to update all currently open animation windows to the simulation step or event immediately preceding the time marked by the Wave window cursor.


A **Cause** command is also added to the **Debug** menu in the simulator List window which can be used to update all open animation windows to the simulation step or event corresponding to the selected line in the List window.

You can also move through the simulation by using a *ModelSim* cursor. The state machine animation and the values displayed in the simulation probes are setup to track the cursor by default.

 These options can be enabled or disabled by using the  button or by setting options in the state diagram **Display** menu and the block diagram **Probes** menu.

A  button is added to the toolbar (and a **Trace To HDS** option is added to the **Debug** menu) in the Source window which can be used to open the graphic window showing the source object corresponding to the line of HDL code under the cursor.


The **Trace To HDS** command is available in the **Debug** menu for the *ModelSim* Wave, List and Signals windows (or using the  button in the Wave window). It can be used in these windows to display the source object where the selected signal is declared. (Typically, this will be the test bench or top level block diagram for the design being simulated.) You can also trace a signal to HDS from the Wave or Signals windows by double-clicking on the signal.

A  button and **Trace To HDS** command are available from the **Debug** and popup menus in the main simulation window. A **Trace To HDS** command is also added to **Debug** menu in *ModelSim* Structure window. These commands can be used to open the source object corresponding to the selected instance.

Refer to “Cross Probing from *ModelSim*” and “Using the *ModelSim* Source Window” in the *HDL Designer Series Graphical Editors User Manual* for more information about the cross-probing commands added to *ModelSim* when it is used with a HDL Designer Series tool.

28. Exit the simulator (by choosing **Quit** from the **File** menu in the main *ModelSim* window). Any other simulator windows are automatically closed.
29. Close all graphic editor windows by choosing **All Editor Windows** from the **Close** cascade of the **File** menu in the design manager.

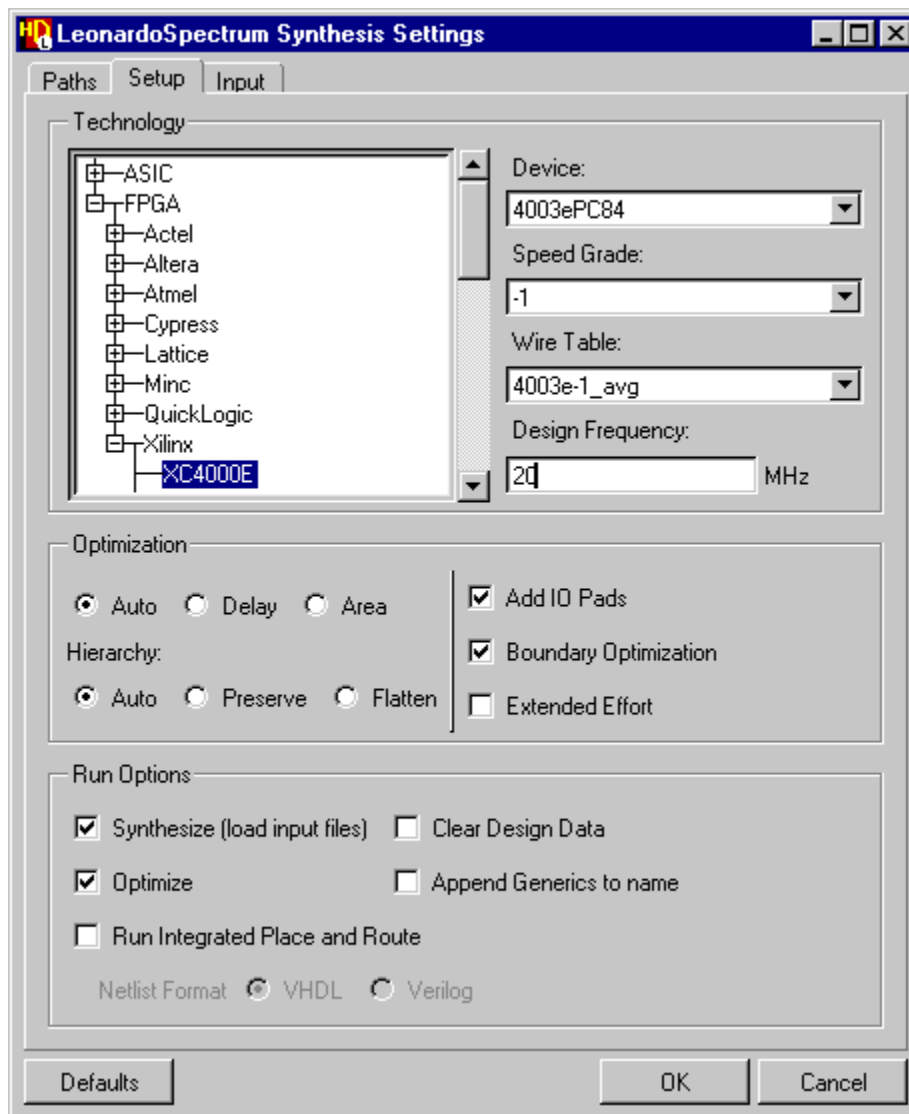
Run the LeonardoSpectrum Flow


1. Select the *Timer* design unit in the design explorer and use the  button to run the LeonardoSpectrum flow through components.



The tester design unit used in the test bench contains unsynthesizable HDL. Ensure that you have selected the *Timer* and not the *Timer_tb* design unit in the design explorer.

The design will be regenerated if necessary and the LeonardoSpectrum Synthesis Settings dialog box is displayed:



- Use the  icons to expand the FPGA technologies list and select the technology of your choice. For example, Xilinx XC4000E.



If you have an Exemplar level 3 license, ASIC and FPGA technology libraries are available. If you have a level 2 license, only the FPGA libraries are available.

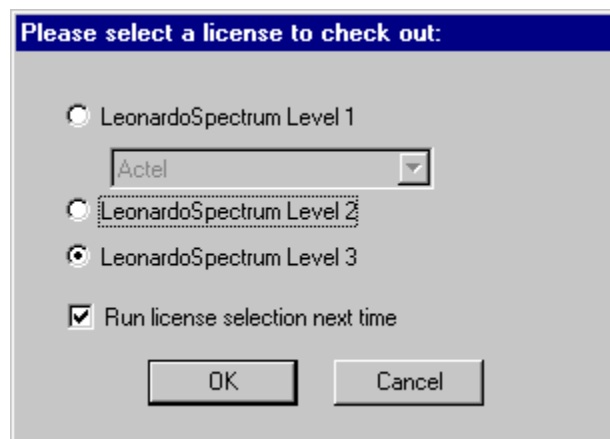
The Device, Speed Grade and Wire Table fields are automatically selected when you have chosen a technology.

- Complete the dialog box by entering a clock frequency (for example 20 MHz). All other options can be left with their default values.

The options in the other tabs can also be left with their default values.

- Use the  button to confirm and close the dialog box.

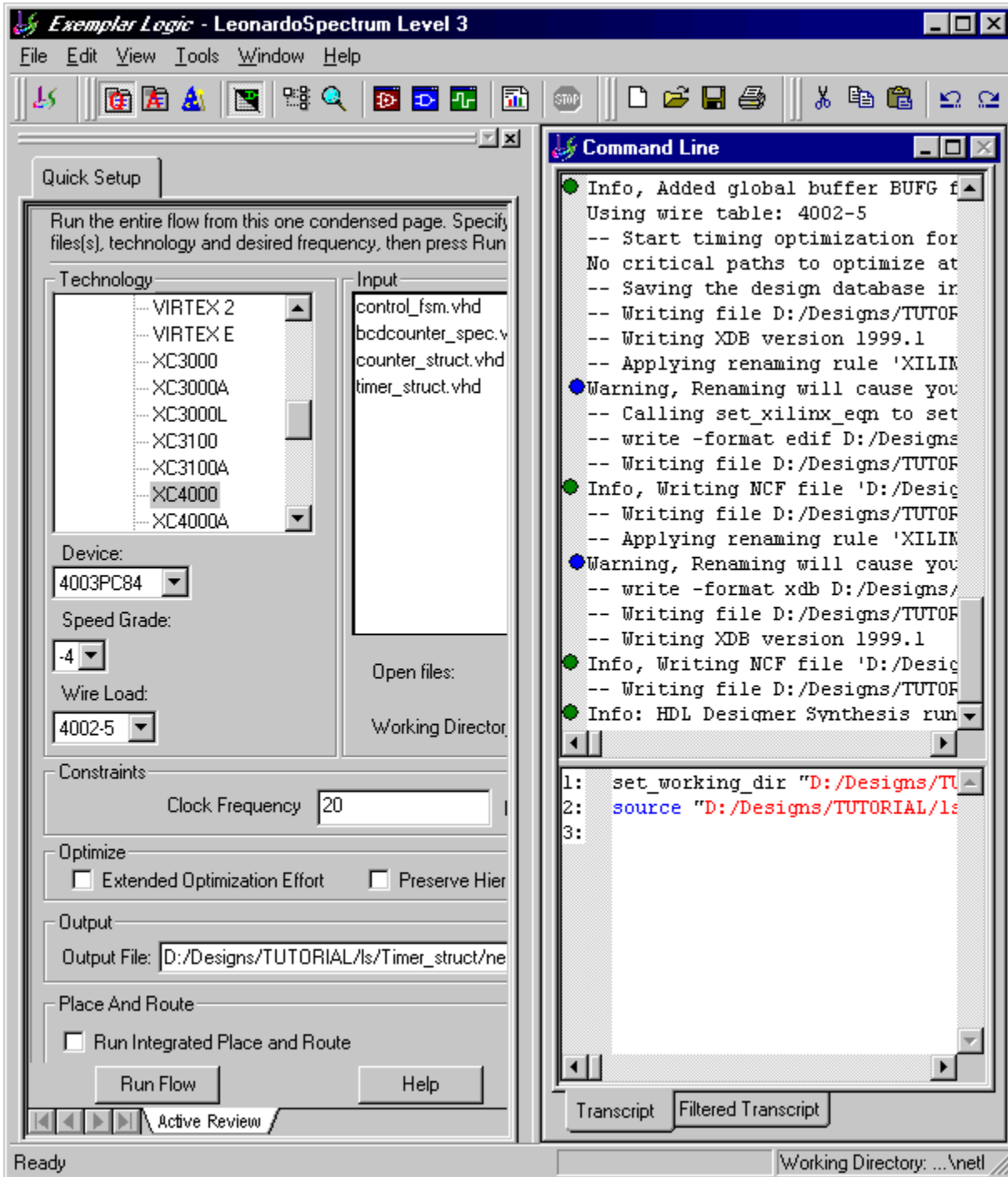
When you confirm the invoke settings, LeonardoSpectrum is invoked and you are prompted to select an Exemplar level 1, 2 or 3 license.





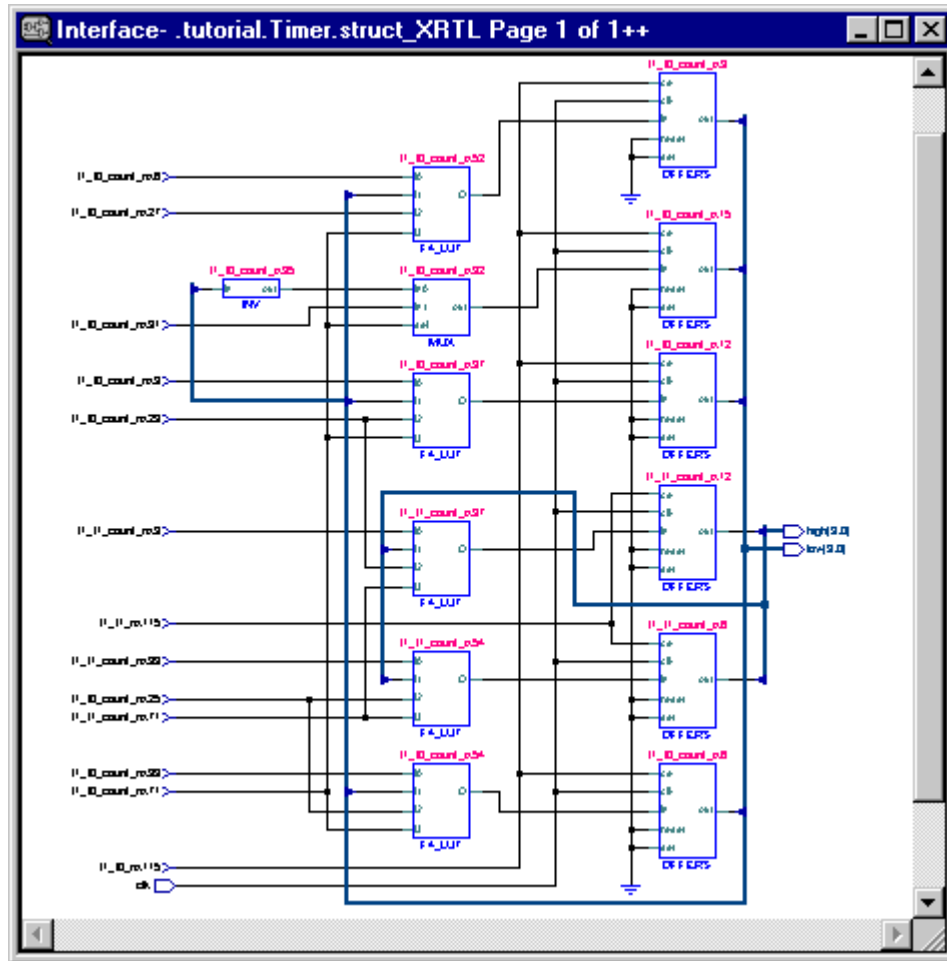
Higher level licenses enable more features but you can choose to run at a lower level using a higher level license.

When you confirm the license, your design is synthesized and optimized for the selected technology.

Status messages are transcribed in the Command window ending with the message “HDL Designer Synthesis run finished”:



You can use the  button from the LeonardoSpectrum toolbar to display the RTL schematic or the  button to display the technology schematic for your design. For example, the following picture shows the RTL schematic:



You can move around the schematic using the scroll bars or the diagram can be enlarged or maximized in the schematic window by choosing the **Zoom In** or **Zoom Fit** options from the **Zoom** cascade of the popup menu.

You can crossprobe to the corresponding source design object by selecting an object in the schematic window and choosing **Trace to HDL Designer** from the popup menu.

- Exit from the synthesis tool by choosing **Exit** from the LeonardoSpectrum **File** menu and respond when you are prompted whether to save your project settings.

You have now completed this tutorial. If you have not successfully verified your design, a completed reference example is available in the examples subdirectory of the HDL Designer Series installation. This example can be accessed by opening the *TIMER_Vhdl* library in the design explorer.

Using the Example VHDL Design

A completed example design can be accessed from the *TIMER_Vhdl* library in the *Examples* project. The example design includes generated HDL but only dummy library mapping for downstream data.

To use the example design, you should change the location of the downstream data directory to a location for which you have write permissions. Refer to Editing Library Mapping in the *HDL Designer Series User Manual* for information about changing library mapping.

Alternatively, you can easily create your own copy of the example design by using the procedure described in Using the Example Designs in the *Start Here Guide for the HDL Designer Series*.



Windows users typically have write access to the installation directories. However, it is recommended that you change the mapping to a suitable directory for user data rather than overwrite the installed examples.

Once you have modified the library mapping, you can browse, compile, simulate and animate the example design by following the procedures from “[Browse the Completed Design](#)” on page 1-72.

In order to animate the flow chart and state machine in the example design, you should regenerate HDL through components after setting **Instrument HDL for animation** in the state machine and flow chart properties. (Refer to “[Set Generation Properties](#)” on page 1-39 and “[Generate HDL for the Test Bench](#)” on page 1-70.)

Chapter 2

Verilog Timer Exercise

This exercise creates a simple timer using block diagrams and a control block described as a hierarchical state machine. A simple truth table is used to decode four-bit binary codes from the ten-bit input bus. The design is completed using a re-usable component described by a HDL text view.

A test bench is created using a flow chart which can be used as a test harness to simulate the generated Verilog for the timer design. The simulation results can be displayed as animation on the flow chart and state machine to assist in debugging the design. The verified timer design is then synthesized.

The tutorial instructions assume that a *ModelSim* or *NC-Sim* simulator and the *LeonardoSpectrum* synthesis tools are available. However, the Verilog generated from the diagrams can also be used by other compatible downstream tools that are available on your system.

Specification

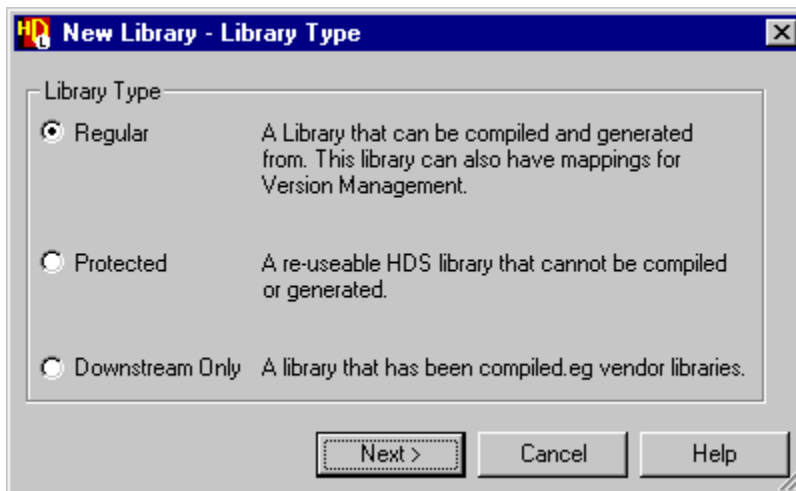
The timer outputs time data on two four-bit buses representing low and high values. There is also a logic output signal which triggers an audible alarm. The data input is provided on a ten-bit bus and control is provided by start, stop, reset and clock signals. These signals are summarized in the following table:


Inputs	Outputs
start (logic signal)	high (4-bit bus)
stop (logic signal)	low (4-bit bus)
reset (logic signal)	alarm (logic signal)
clk (logic signal)	
d (10-bit bus)	

Set Library Mapping

For this tutorial, you can use an existing project such as the project you created for the Design Management tutorial but should create a new library mapping for your design data. The library mapping defines the logical location of the directories containing your design data. The source graphical objects, HDL and downstream data can be stored at any writable locations on your available file system but are typically saved below a common root directory.

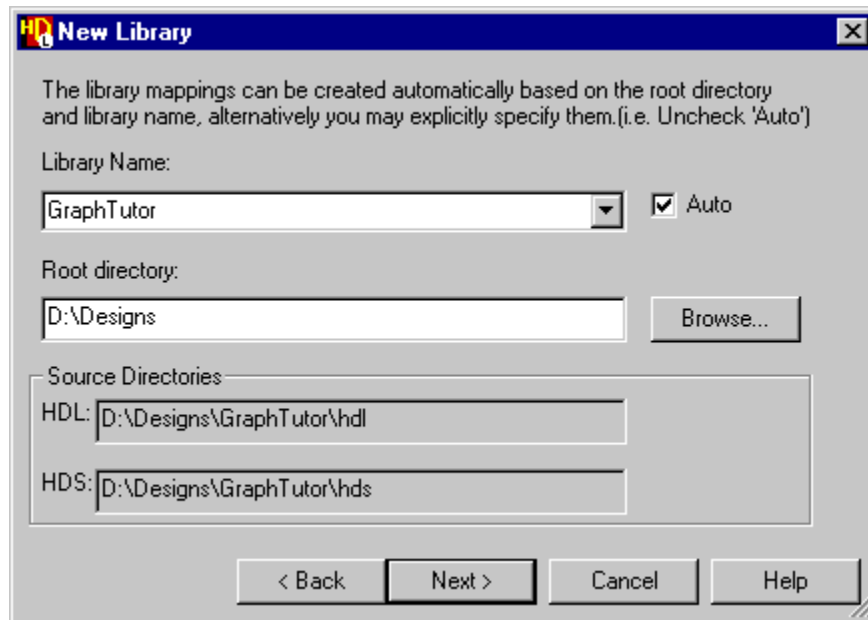
1. To set library mapping, click the  **New Library** icon in the Project group on the shortcut bar or choose **Library** from the **New** cascade of the **File** menu in the design manager window to display the New Library wizard:




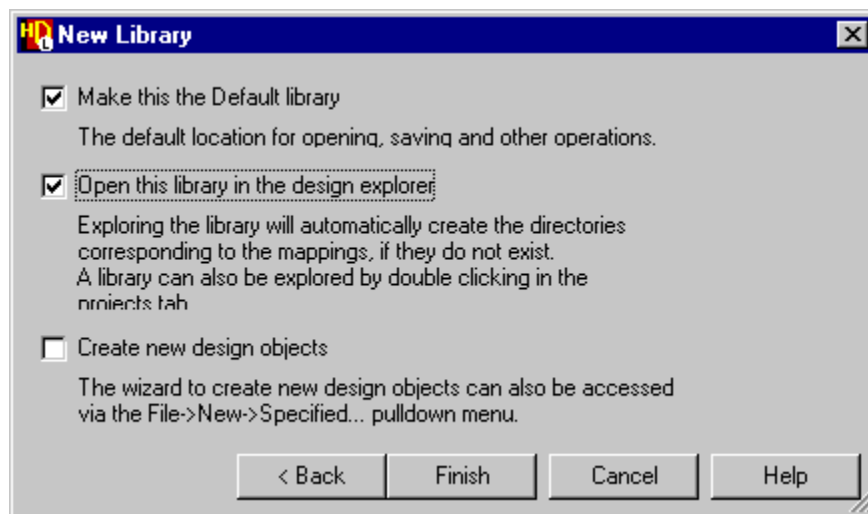
2. Choose the **Regular** library type and click the  button to display the library name page.
3. Enter a logical library name (for example: *GraphTutor*) and check that the **Auto** option is set in the dialog box.
4. Specify or browse for the pathname for the root directory that will contain your library data (for example, *D:\Designs* or *\$HOME/designs*).


Library names and pathnames can be entered using upper, lower or mixed case but note that UNIX systems are case sensitive and the case used for pathnames should match the file structure. (On a PC, library names are case sensitive but pathnames are case insensitive.)

Notice that the source HDL directory is named `<root_directory>\GraphTutor\hdl` and the source graphics directory is named `<root_directory>\GraphTutor\hds`.

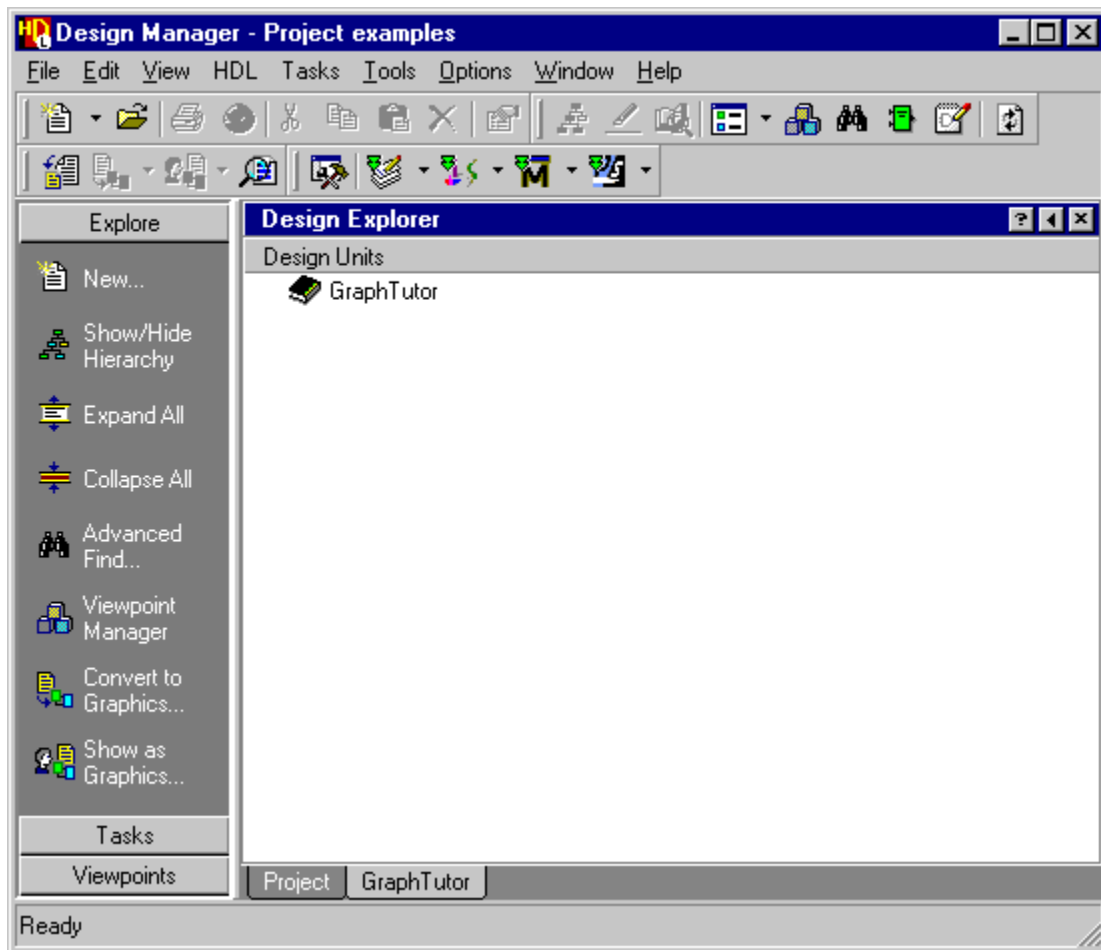


5. Click the  button to display the last page of the wizard.



6. Set the **Make this the Default library** and **Open this library in the design explorer** options.
7. Click the  button to close the wizard.

The source design data directories you specified in the wizard are created and the library (shown as a closed “book”) is opened in a new design explorer window:

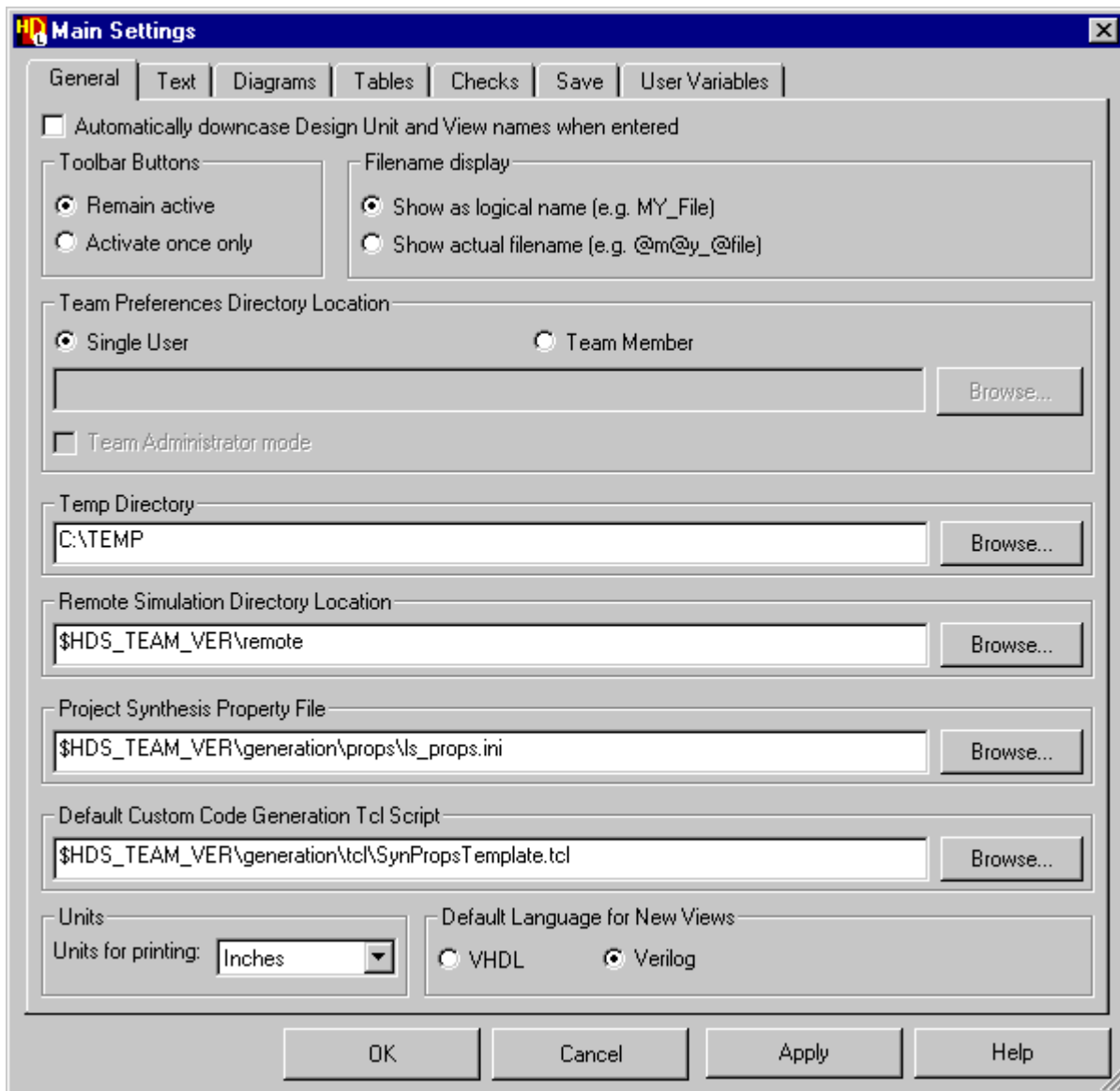


The mapping for the location of compiled data will be defined automatically when the design is compiled.

Set the Default Language

A set of default preferences are loaded when you invoke a HDL Designer Series tool for the first time. There are separate tabbed dialog boxes for the main settings, VHDL and Verilog options, HDL2Graphics import options, version management, animation settings and master preferences for each type of graphical diagram. The preference dialog boxes can be accessed from the **Options** menu.



1. Choose **Main** from the **Options** menu to display the Main Settings dialog box.

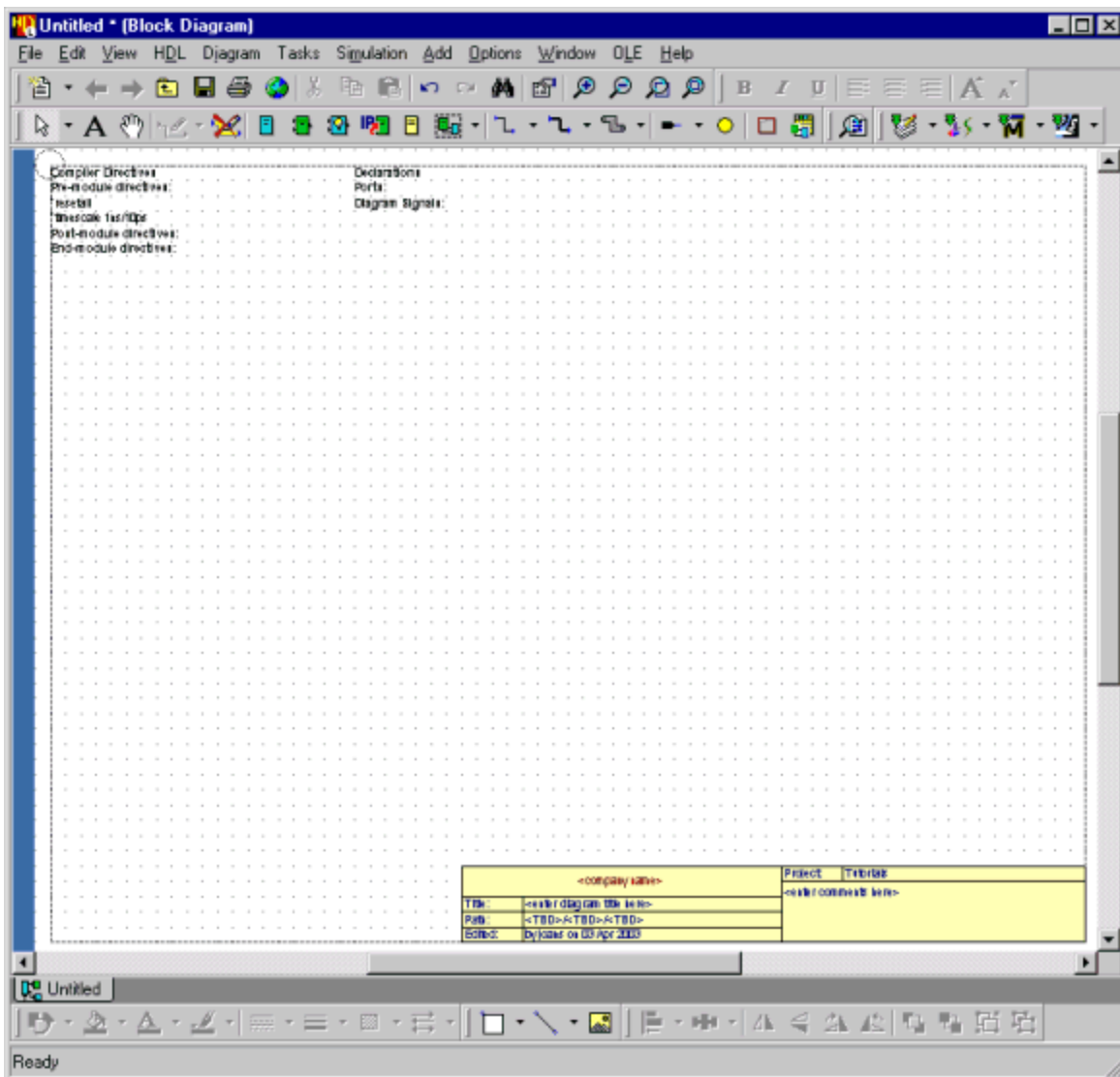


2. Select the **General** tab and ensure that **Verilog** is set as the default language to be used for new graphic editor views. Use the **OK** button to confirm your choice.

All other preferences can be left with their default values for this tutorial.

Create a Block Diagram

1. Use the  button in the design manager window and select **Block Diagram** from the **Graphical View** cascade of the dropdown menu to create a new untitled block diagram.
2. Use the  button to view the entire diagram:



Notice the five toolbars at the top and three toolbars at the bottom of the diagram. The toolbar buttons provide quick access to many of the most frequently used editing and formatting commands.

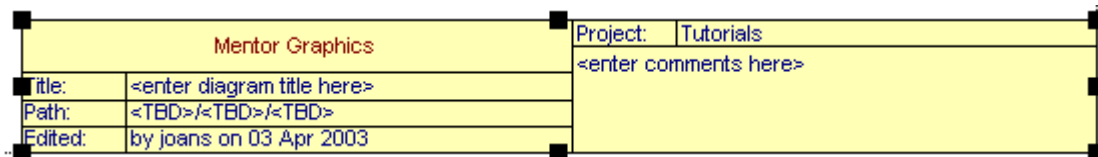
The block diagram is a blank sheet except for a background grid, a list of compiler directives (with *'resetall* and *'timescale 1ns/10ps* set by default) and empty text fields with labels for *Declarations*, *Ports* and *Diagram Signals*.

The diagram also shows page boundaries for the default printer and a copy of the default title block.

Edit the Title Block Template

A title block is automatically added to all new diagrams if the **Add Title Blocks in new diagrams** option is set in your diagram preferences. The default title block is a template with default locations for your company name, diagram title and comments. The template incorporates internal variables which automatically enter the current project name, your login name and the current date. Internal variables are also used to enter the logical pathname for the design. This path is initially shown as `<TBD>/<TBD>/<TBD>` but the internal variables are converted to show the library, design unit and view name when you save the diagram.

- Click twice on `<company name>` in the default title block to display a popup edit box and replace the default text by the name of your company.
- Select the title block by clicking with the mouse so that the selection handles are displayed and choose **Save Title Block** from the **File** menu.



- A dialog box is displayed with a warning that saving the title block cannot be undone. Click the button to proceed.

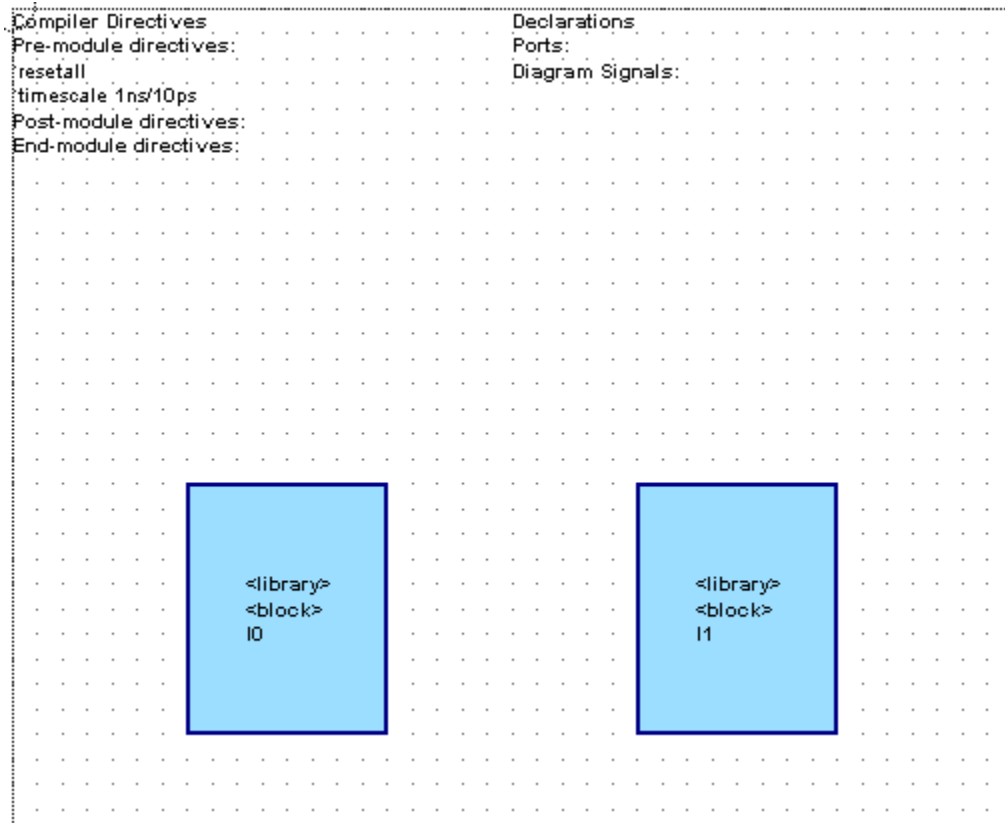


The title block is saved as a team resource at the location specified in your preferences and will be used as the default template in new diagrams. If shared team preferences have been set up on your system, there may already be a read-only team template stored at this location.

Refer to the *HDL Designer Series User Manual* for information about how you can add and modify title blocks.


Add Blocks




6. Use the  button (or choose **Block** from the **Add** menu) to add two blocks on the diagram as shown below:




Notice that the blocks are added with the default library *<library>*, the default name *<block>* and unique instance names (*I0* and *I1*).

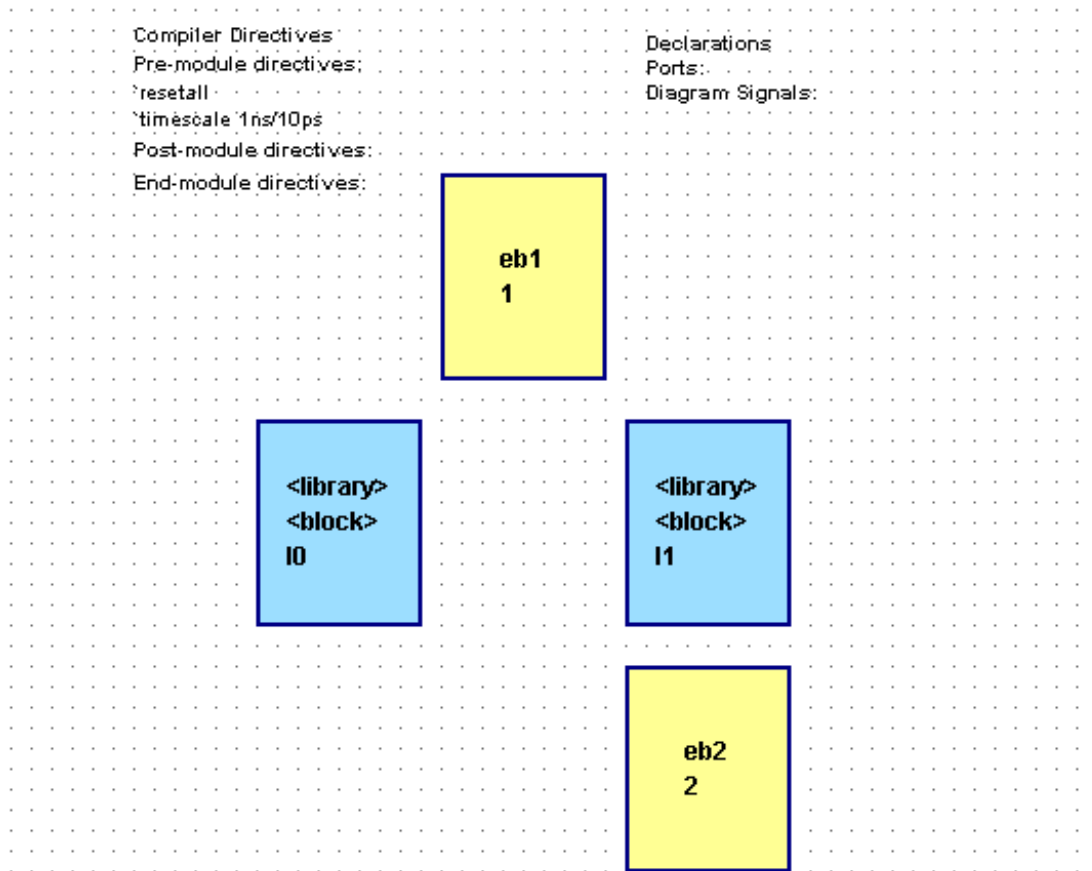


The Add Block command normally auto-repeats until you select another command or terminate the repeating command by using the right mouse button or  key. However, you can change the behavior of the toolbar buttons by setting the **Activate once only** preference in the **General** tab of the Main Settings dialog box.

You can also use the  key with any toolbar button to toggle the repeat mode. For example, when **Remain active** is set,  +  adds a single block on a block diagram.

Add Embedded Blocks

7. Use the  button to add two embedded blocks on your block diagram as shown in the picture below:








Notice that the embedded blocks are added with unique default names (*eb1* and *eb2*) and numbers (*1* and *2*)


The view describing a block must be saved as a uniquely named design unit in a library directory. However, the view describing an embedded block is saved as part of the parent block diagram and does not impose hierarchy when HDL is generated for your design. The name of an embedded block must be unique on the diagram and is used as a label in the generated HDL.




The blocks (*I0* and *I1*) will be used to define a child state machine and block diagram view. The embedded blocks (*eb1* and *eb2*) will be defined by concurrent assignment statements on the top level block diagram.


Add Ports and Signals

You can use the following buttons to add signal and bus nets on a block diagram:


- | | |
|---|--------------------------|
|  | Add a signal |
|  | Add a signal with a port |
|  | Add a bus |
|  | Add a bus with a port |
|  | Add a bus with a ripper |

Signals or buses can be added between any existing connectable items on the diagram or left unconnected by double-clicking to terminate the net with a dangling net connector. However, you can use the  pulldown on the buttons to change the default setting and terminate with a default port or ripper. Notice that the toolbar button changes to show the current setting.


When the  or  button is selected, a port is automatically added at an unconnected source or destination end point. When the  button is selected, a ripper is used if the end point is over an existing bus or bundle.

- Choose **Signal with Port** and use the  button to connect three signals originating from the block on the left (instance *I0* in the picture) to the block on the right (instance *I1*) and one signal returning from *I1* to *I0*. The signals are added with unique names (*sig0*, *sig1*, *sig2* and *sig3*) and the default type *wire*. Notice how the full declarations for these diagram signals are automatically added to the list of Declarations on the diagram.

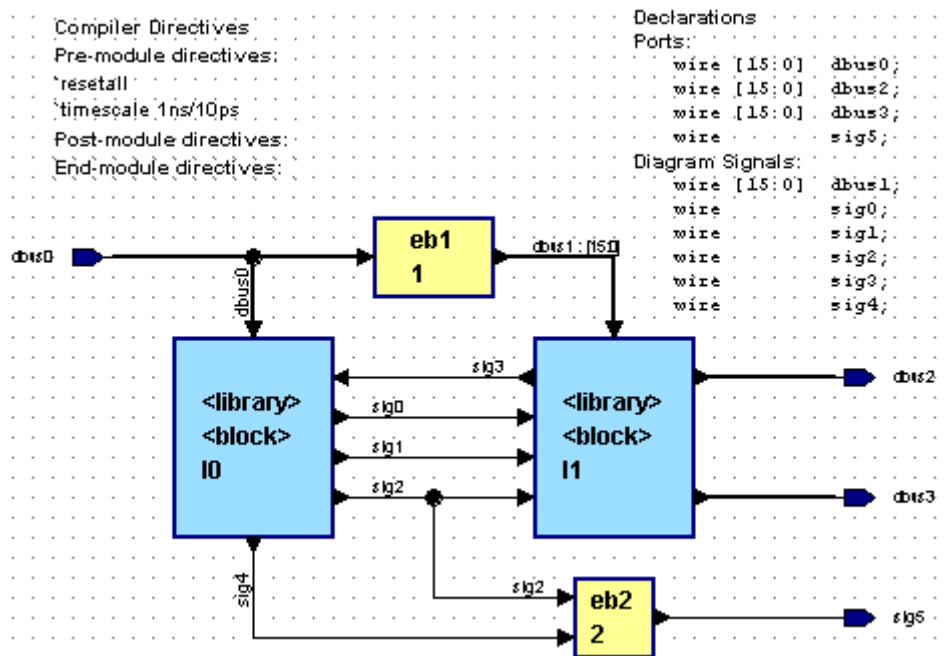


Allow one or more grid lines between each port or signal. You can resize objects by selecting a block or embedded block and dragging one of its resize handles. If necessary, you can drag text elements such as the signal name and type using the  mouse button.




- Add a signal from *I0* to the embedded block *eb2* and another signal from a point on *sig2* terminating on the embedded block.

10. Add a signal from the embedded block *eb2* terminating in space on the right side of your diagram. Notice that an output port is added when you double-click at the end of the last signal and its declaration is added to the list of ports on the diagram.
11. Choose **Bus with Port** and use the  button to add a bus from a source on the left side of your diagram with its destination on the upper embedded block *eb1*. A default input port is automatically created at the beginning of the bus.
12. Add another bus starting from this bus and terminating on instance *I0*. Notice how both bus segments have the same default name *dbus0*. The full declaration showing the default bus type and bounds *wire [15:0]* is added to the list of ports.
13. Add a bus (*dbus1*) from *eb1* to *I1*. This internal diagram signal is shown with the default bounds *[15:0]* shown on the net.
14. Then add two buses (*dbus2* and *dbus3*) from *I1* terminated with default output ports on the right of the diagram.





Your diagram should now look similar to the picture below:





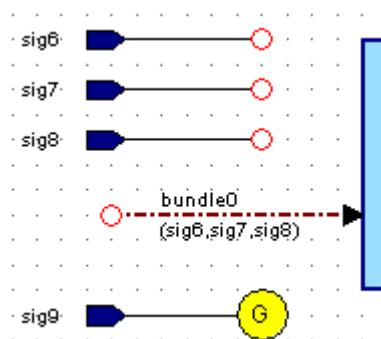
Add a Bundle and Global Connector




- Use the  button to add three signals on the left side of your diagram. Notice that a default input port is added at the source of each signal but a dangling net connector is drawn when you double-click at the end of each signal.
- Select the three signals (by dragging a select rectangle with the  mouse button held down) and use the  button to connect a bundle containing these signals to block instance *I0* as shown in the picture below.

Notice that the bundle has the default name *bundle0* and the three selected signals are automatically included in the bundle with their names listed under the bundle name.

 You can use the  pulldown on the  button to select a  button which allows you to rip one or more signals and buses from an existing bundle.


- Use the  button to add a global connector on your diagram below the bundle.
- Use the  button to add a signal between the global connector and a default input port. (This will be a clock signal which is implicitly connected to every block on the diagram.)

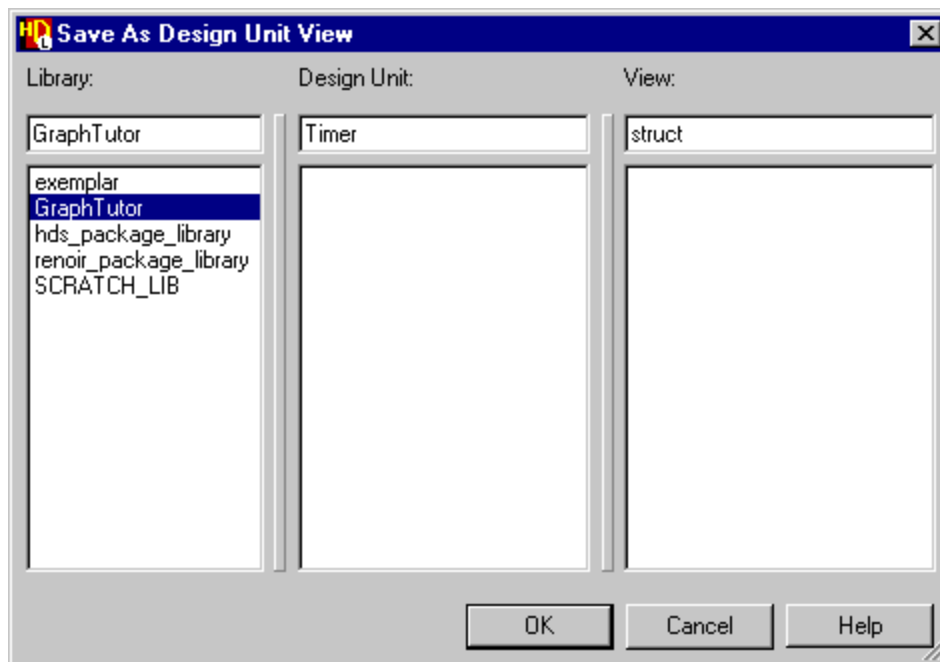


 If you make a mistake when editing a diagram, you can use the  button to undo your last edit and the  button to redo an undo operation. You can also use commands from the **Align** cascade of the **Edit** menu to re-align and distribute objects on the diagram.

Save the Block Diagram

Notice the asterisk (*) character in the header of the block diagram editor window. This indicates that the diagram has been edited since it was last saved.

19. Use the  button to save the block diagram. The Save As dialog box is displayed which allows you to choose from the currently mapped libraries and specify the design unit and design unit view names.
20. Choose the *GraphTutor* library and enter design unit *Timer*. The dialog box should look similar to the example below:



The view name and file extension when you save graphical diagrams defaults as follows:

struct.bd	block diagram
struct.ibd	interface-based design view
fsm.sm	state diagram
flow.fc	flow chart
tbl.tt	truth table
symbol.sb	symbol



If you omit the two-character extension it is automatically added to identify the type of diagram you are saving. The default leaf names can be changed by setting preferences. However, you should not change the extension (*.bd*, *.ibd*, *.sm*, *.fc*, *.tt* or *.sb*) or the design data file will not be recognized and cannot be reopened.

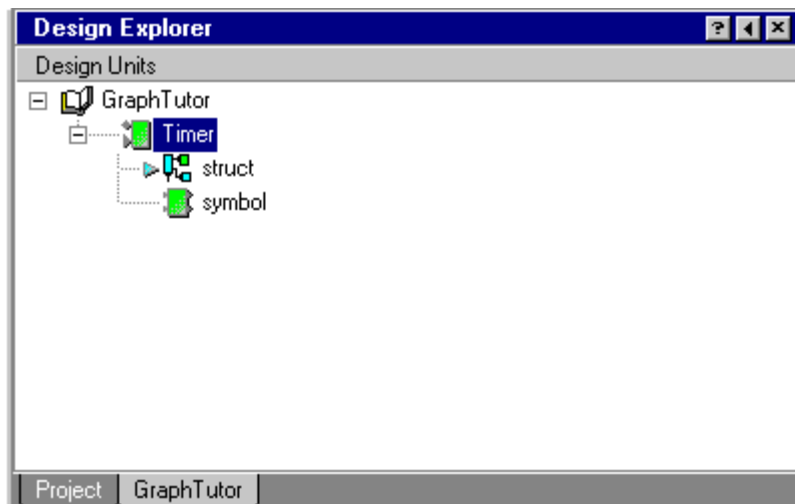
21. When you click the button, your diagram is saved and the window title bar is updated to show the diagram name *GraphTutor\Timer\struct*.

This path is also added in the title block replacing the <TBD> used when the diagram was created. Notice that the asterisk (*) character has been cleared in the block diagram header and the library name used on the blocks in your diagram has been updated to *GraphTutor*.



If the design manager window is obscured, you can pop it to the front by selecting **Design Manager** from the list of open windows in the block diagram **Windows** menu.

22. Click on the  icon for the *GraphTutor* library in the design explorer and notice that the view is expanded to display the *Timer* design unit.
23. Click on the  icon for the *Timer* design unit to reveal that it contains a symbol and block diagram view.



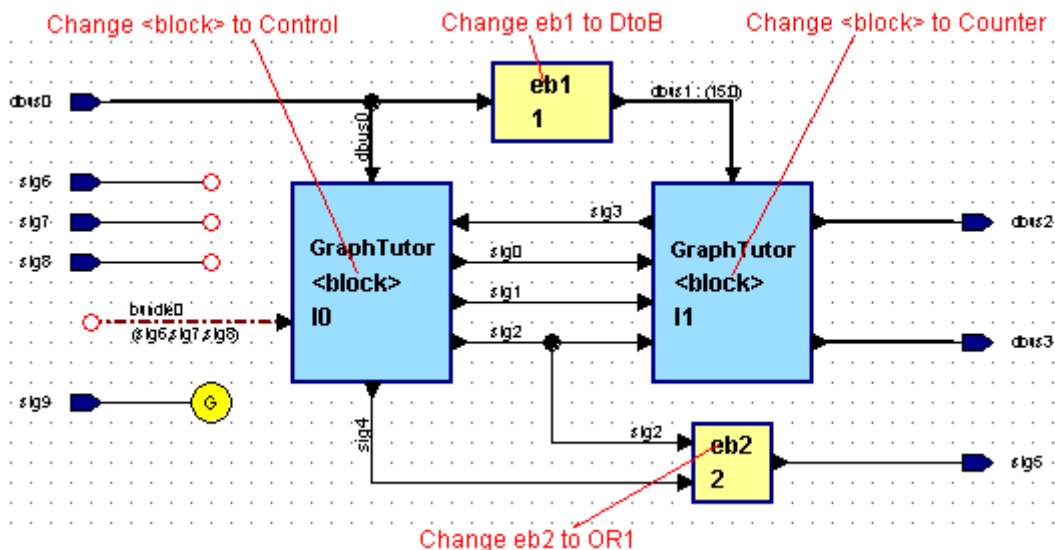
Edit Block and Signal Names

You now have a completed top-level block diagram for the *Timer* design. However, the blocks and signals have default names.

- Click on the text `<block>` in the lower block on the left (instance *I0* in the picture) and notice the small handles which indicate that the text object is selected. Click again and notice that the text is now highlighted and can be directly overwritten.


If you click again, the cursor changes to an I-beam which allows you to move the cursor in the text and edit individual characters. Enter the new name *Control* and click outside the text to complete the edit.

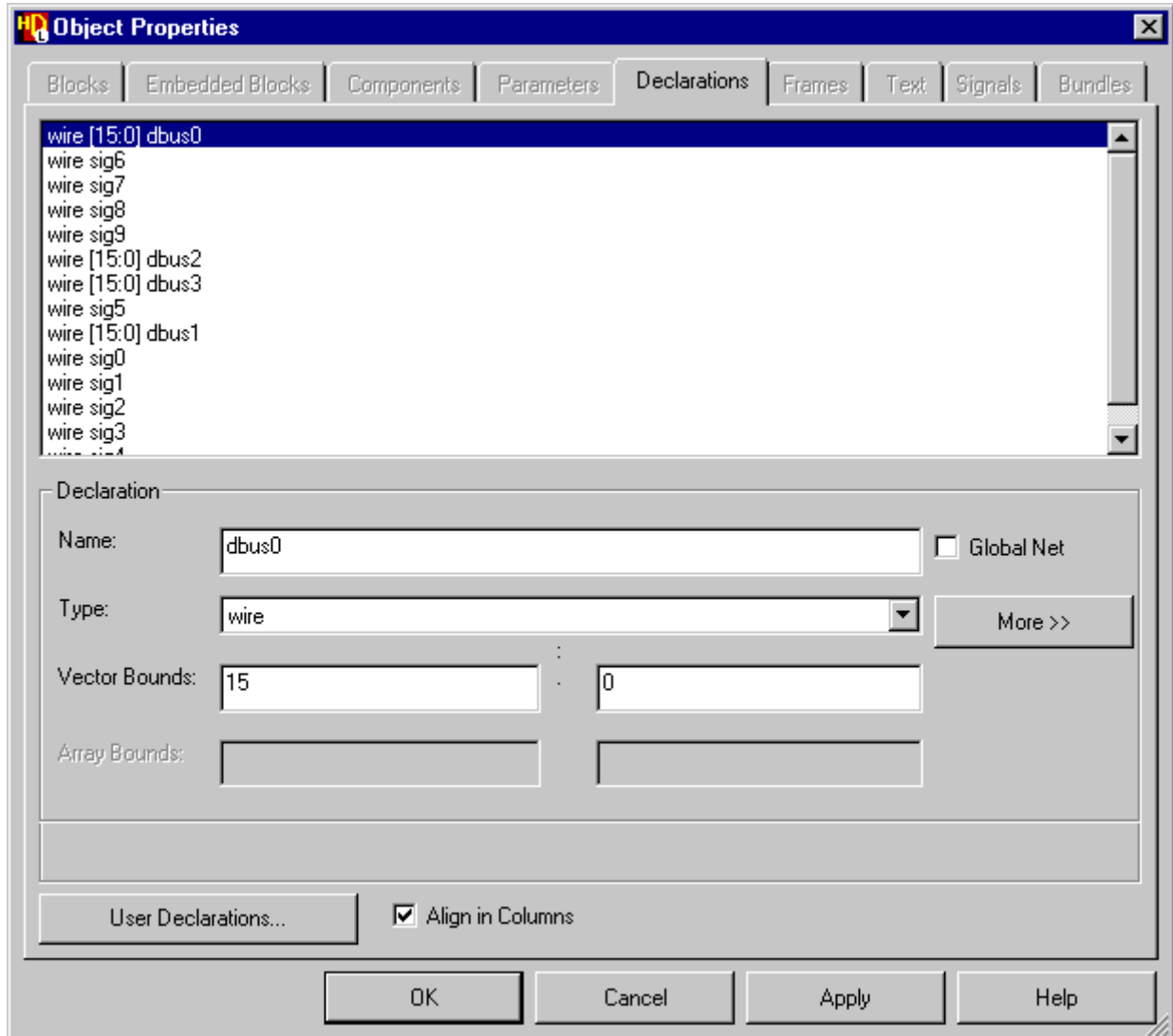
- Repeat this procedure to change the name of block instance *I1* to *Counter*, embedded block *eb1* to *DtoB* and embedded block *eb2* to *OR1*.





Direct text editing can also be used to edit the signal and bus names. Alternatively, you can use a dialog box which allows you to edit the properties for a selected object. By default, edits to signal and bus nets are applied only to the connected nets but you can choose to apply the changes to the entire diagram or to propagate changes to all occurrences of the net in the hierarchy of the design.



- Choose **Entire Net in diagram** from the **Scope for Changes** cascade of the **Signals** cascade in the **Diagram** menu.

27. Double-click on the existing declarations, use the  button or choose **Object Properties** from the **Edit** menu to display the Block Diagram Object Properties dialog box and choose the **Declarations** tab.



Notice that the port declarations are listed at the top of the dialog box and the other internal diagram signals at the bottom. Input ports are listed before the output ports, otherwise the declarations are listed in alphanumeric order.

You can choose one or more existing declarations in the dialog box, enter new values for any of the declaration fields. For example, use  +  mouse button to choose *dbus1*, *dbus2* and *dbus3*, then enter a new bounds *3:0* to update all three buses while all other fields remain AS IS.

The changes are applied to the diagram when you click the  or  button. Notice that all occurrences on the diagram are updated including the declarations list, signals, buses and bundle contents and that the lists of port and signal declarations are sorted alphanumerically when the dialog box is applied to the diagram.

28. Use the dialog box to update the port and signal declarations as shown in the following tables.

Ports:

Old Name	New Name	Type	Bounds
dbus0	d	wire	9 : 0
sig6	start	wire	none
sig7	stop	wire	none
sig8	reset	wire	none
sig9	clk	wire	none
dbus2	low	wire	3 : 0
dbus3	high	wire	3 : 0
sig5	alarm	wire	none

Diagram Signals:



Old Name	New Name	Type	Bounds
dbus1	dat_in	wire	3 : 0
sig0	clear	wire	none
sig1	load	wire	none
sig2	hold	wire	none
sig3	zero	wire	none
sig4	beep	wire	none



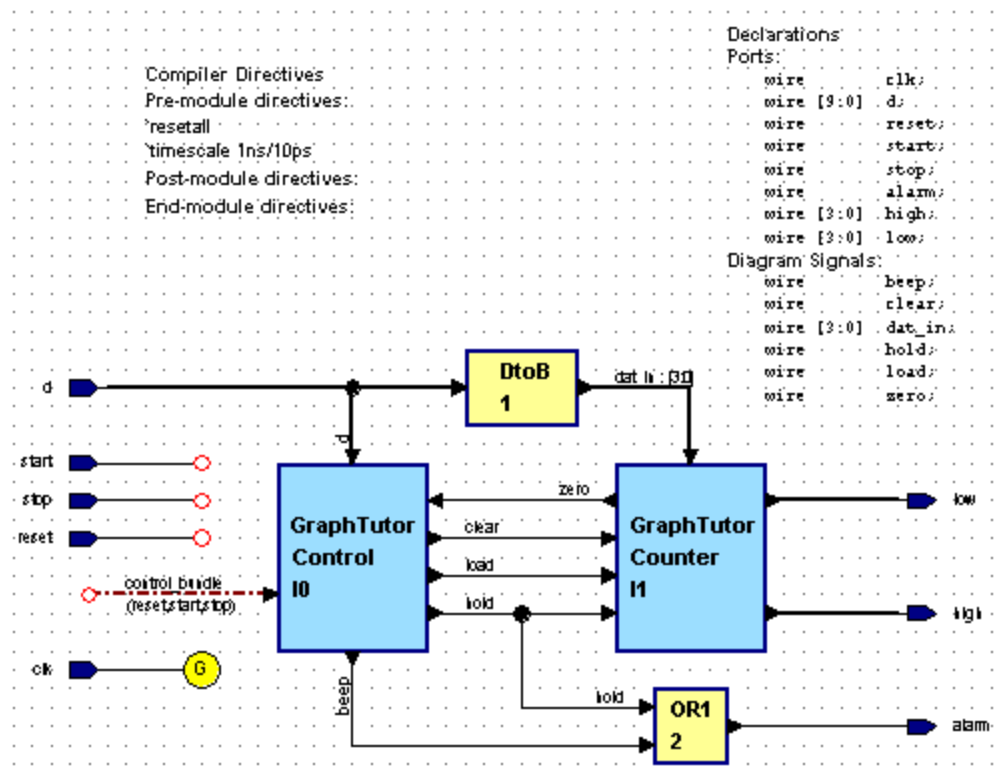
All occurrences of each signal name (including the bundle contents) should be automatically updated on the diagram when you confirm the dialog box. If any nets are not updated, check that you have set the scope for changes to **Entire Net in Diagram** as described on [page 2-15](#).



29. Select the bundle name and use direct text editing or the **Bundles** tab of the Object Properties dialog box to change the bundle name to *control_bundle*.



If you have difficulty selecting text objects, you can change the selection mode by using the  pull-down on the  button to select text or object shapes only.

Your block diagram should now look similar to the following picture:

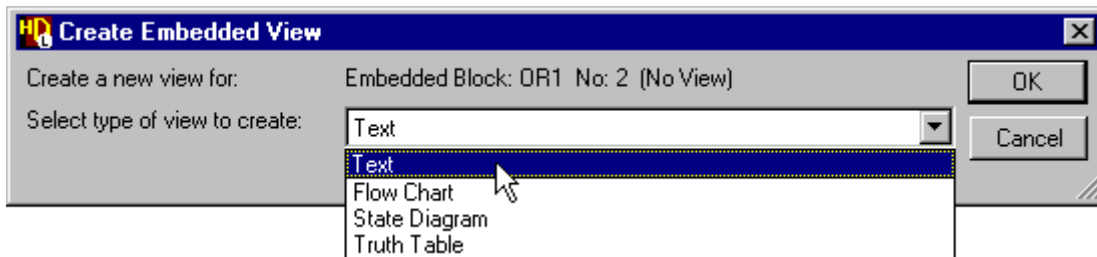


The  button on the dialog box allows you to disclose additional fields which allow you to modify other signal properties including delay, expansion, charge strength, attributes and comments. The  button allows you to add additional user-entered module declarations to the structural Verilog.


Refer to the *HDL Designer Series Graphical Editors User Manual* for more information about these features which are not used in this tutorial.

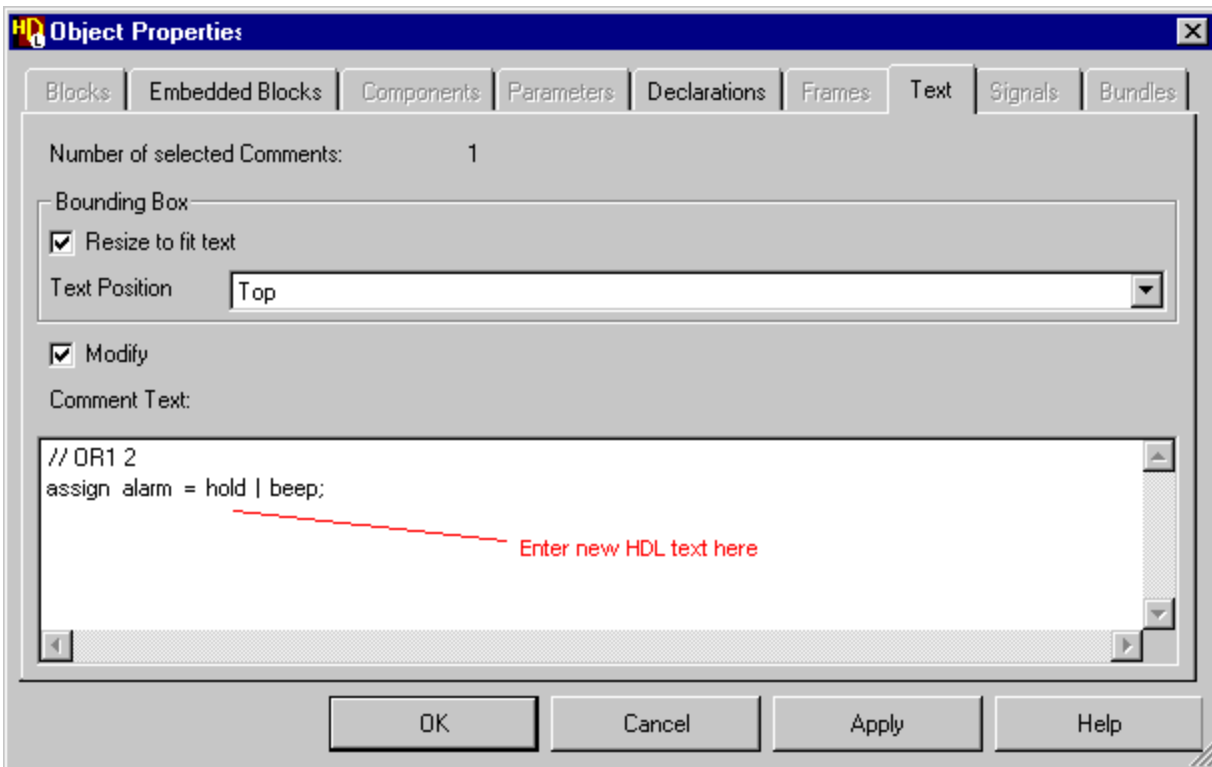
Add an Embedded HDL Text View

30. Select the *ORI* embedded block and display the popup menu by clicking the **Right** mouse button. Choose **New View** from the **Open** cascade in the popup menu to display the Create Embedded View dialog box. Choose Text from the pulldown list of views in the dialog box.



When you confirm the dialog box, an embedded HDL text view containing default text is displayed on the block diagram adjacent to the embedded block.

31. Select the text, re-display the Object Properties dialog box if necessary (using the  button) and choose the **Text** tab.



32. Check the bounding box **Resize to fit text** option and enter the following Verilog statement under the default `// OR1 2` comment text in the dialog box:

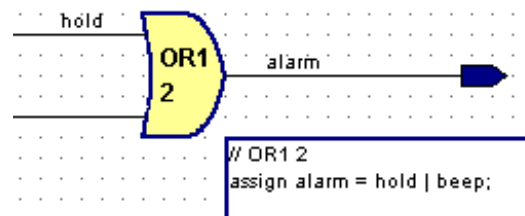
```
assign alarm = hold | beep;
```

The modified HDL text is checked for syntax errors and applied to the diagram when you click the or button on the dialog box.

The functional blocks on the diagram are shown by default as simple rectangular shapes. However, it is sometimes useful to use logic notation when a block has a specific logical function. For example, in this block diagram, the *ORI* embedded block represents a logical OR function. You can apply a logical OR shape using the Comment Graphics toolbar which is normally docked at the bottom of the editor window, by choosing **Autoshapes** from the **Shape** cascade of the popup menu or by using the button in the **Embedded Blocks** tab of the Object Properties dialog box.

33. Select the embedded block and choose the pulldown on the button in the Comment Graphics toolbar to display a palette of alternative shapes. Select from the palette to apply a logical OR shape to the embedded block.
34. Hide the port arrow heads by clearing the **Show Ports when connected** check box in the **Embedded Blocks** tab of the Object Properties dialog box.



The *ORI* embedded block should now look similar to the following picture:



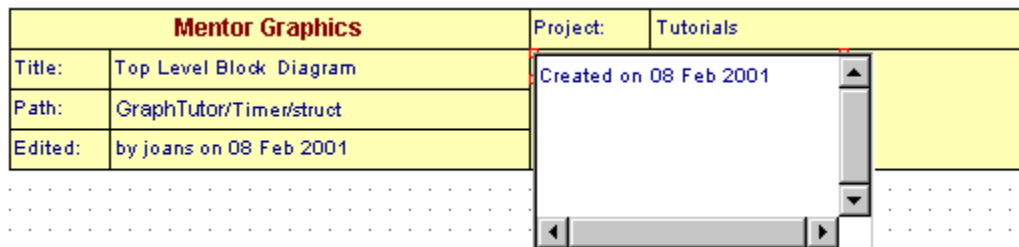
It is also possible to indicate an active low (Not) or edge triggered clock signal. This feature can be used with the alternative shapes to represent extra functions such as an inverter, NAND, NOR or flip-flop. The logical OR function could also be implemented by a ModuleWare part similar to that used in [“Add ModuleWare Components”](#) on page 2-53.

Add a Panel and Edit the Title Block


A panel can be useful to outline areas of a diagram. For example, you can use a panel to outline a view used for simulation or animation.


35. Use the  button to add a panel and hold down the  mouse button to drag the panel and enclose the graphical objects on your diagram. The panel is added with the default name *Panel0*.
36. Complete the block diagram by editing the title and comments in the title block on the diagram. For example, enter the title `Top Level Timer Block Diagram` and a comment of the form: `Created by <your name> on <date>`.


The title block comprises a number of grouped comment text objects. Each comment text object can be edited directly by clicking twice on the text to display a text entry box.



You can enter free-format text including line breaks and spaces which will be preserved on the diagram.

37. Click the  mouse button outside the entry box to complete the text entry.

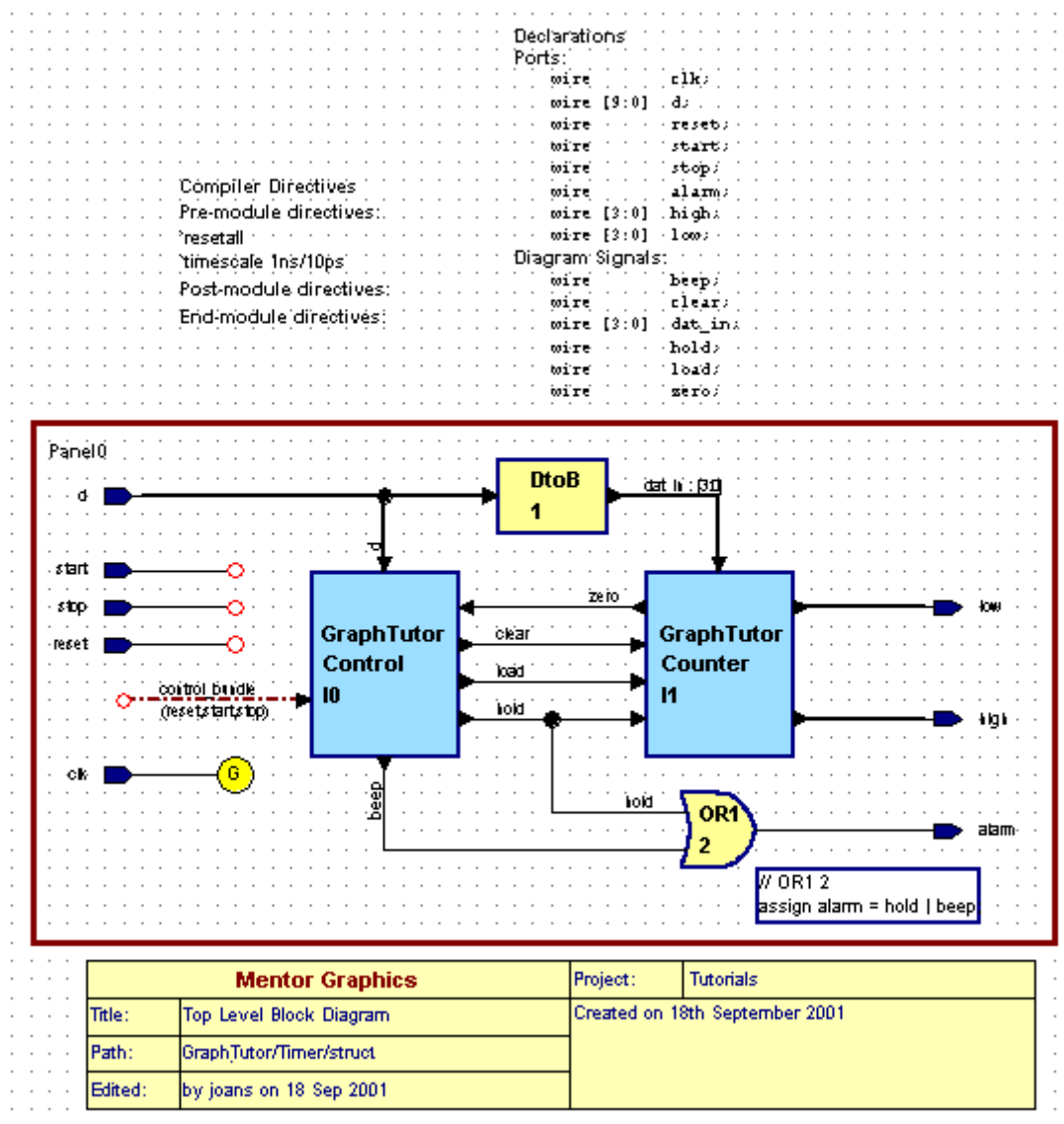
 You can also edit an existing comment text object by double-clicking to display the **Text** tab in the Object Properties dialog box. When comments are edited in the dialog box, it is possible to enter any special characters (for example accents or Kanji characters) which are supported on your system.

38. Use the  button to save the block diagram.

You have previously saved the diagram so you are not prompted for library and design unit names. However, you have changed the names of signals connected to input and output ports and the block will be inconsistent with the symbol that was automatically created by the previous save. You are prompted whether to update the symbol.

39. Click the button to confirm the save.

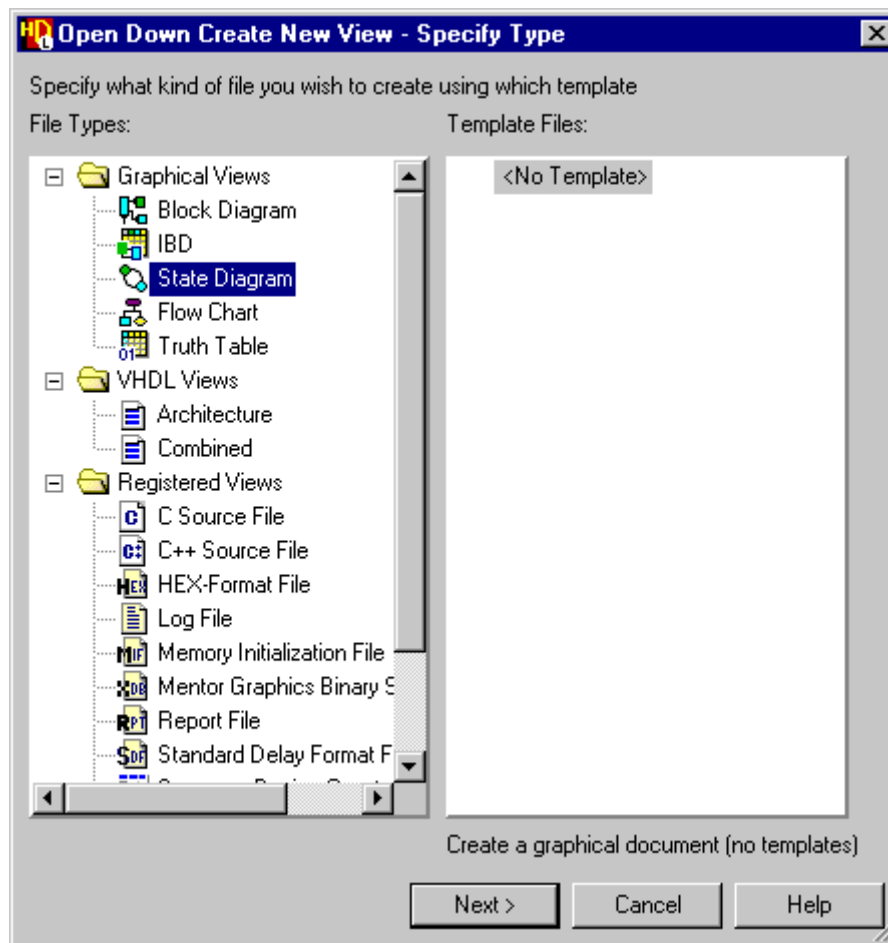
The completed block diagram should look similar to the following picture:




Create a Child State Diagram

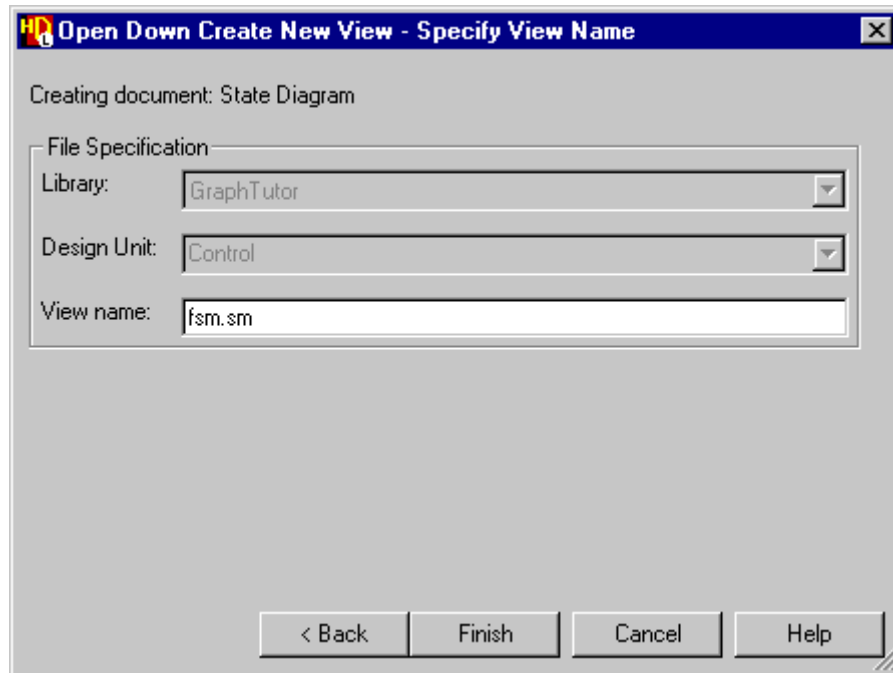
The procedures in the following sections create a graphical state machine to describe the *Control* block.

1. Move the cursor over the body of the *Control* block on the *Timer* block diagram, then press and release the **Right** mouse button to select the block and display the popup menu.
2. Choose **New View** from the **Open As** cascade menu. The Open Down Create New View wizard is displayed:



Solid handles are displayed when the body of a block (or other re-sizable object) is selected. You can display the wizard directly by double-clicking on the body of a block which has no views defined.


3. Select *State Diagram* from the list of file types and use the  button to display the Specify View Name page of the wizard:



The library and design unit fields are shown dimmed because they are copied automatically from the library and design unit of the parent diagram. The view name defaults to *fsm.sm* for a state machine.



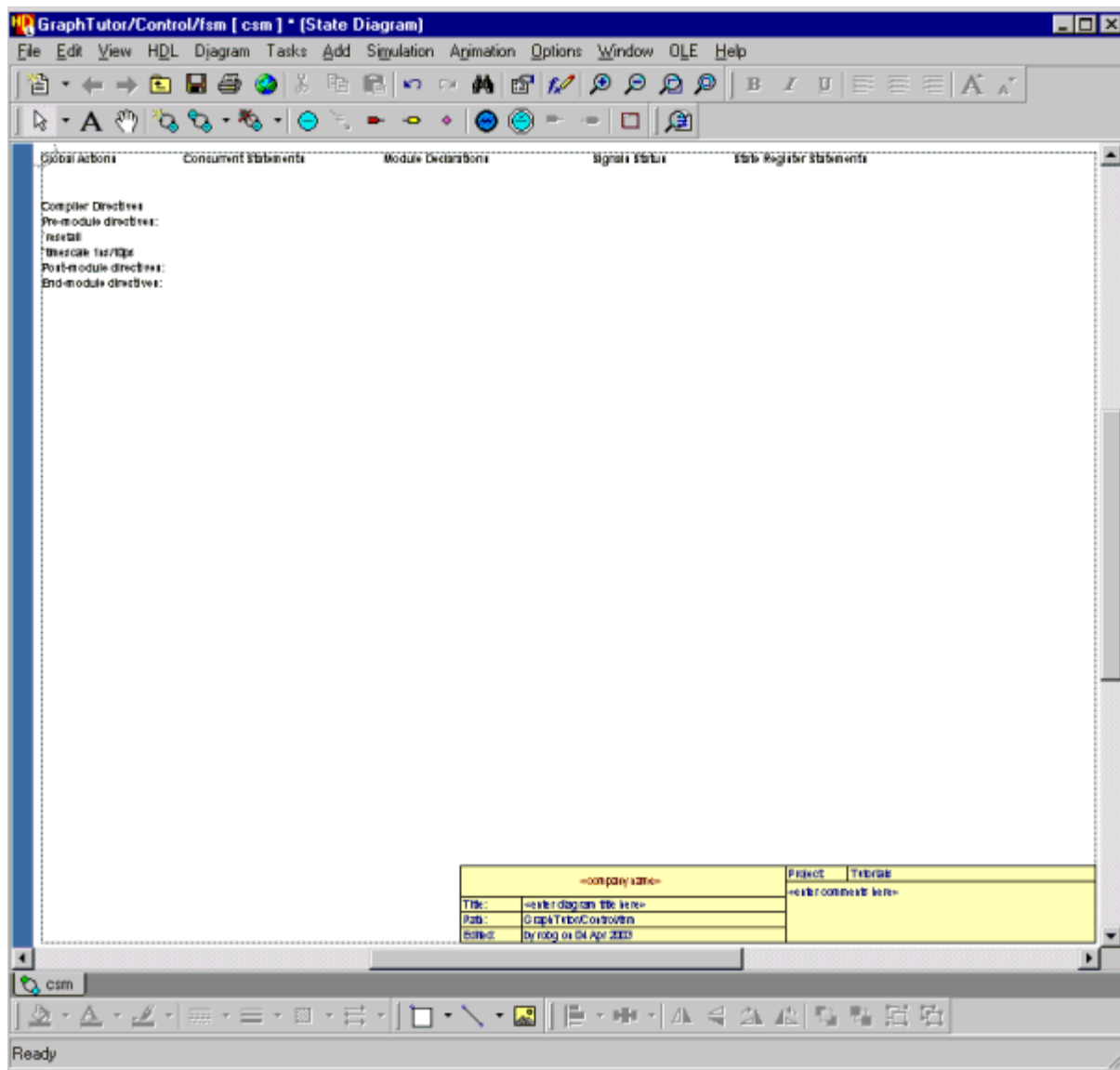
You do not need to enter the file extension (*.bd*, *.fc*, *.ibd*, *.sm* or *.tt*) for graphical views as the correct extension is automatically added. However, if you do enter any other extension you are warned that the file will not be recognized.

4. Use the  button to confirm the wizard with the default view name *fsm.sm*.

A new state diagram *GraphTutor/Control/fsm [csm]* is created as a child view of the *Control* block.


Note that a default state machine name *csm* is appended to the design unit and view names in the diagram title. This name can be changed by choosing **Rename Concurrent State Machine** from the **Diagram** menu in the state diagram.

The state diagram is a blank sheet with page boundaries set for the default printer. The diagram includes text objects for the default compiler directives list and labels for global actions, concurrent statements, module declarations, signals status and state register statements:




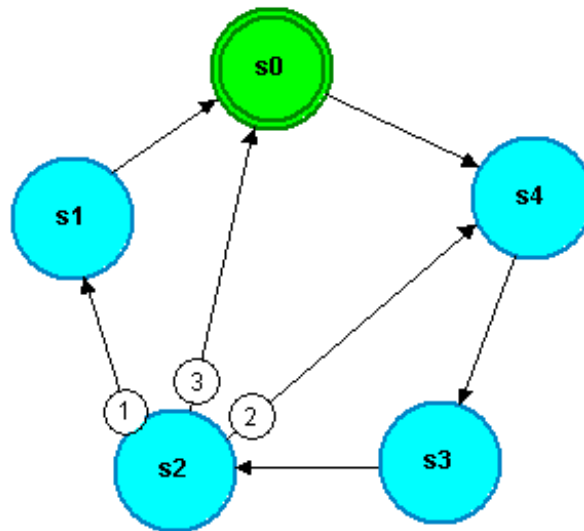
The diagram also includes the default title block which you saved as a template in an earlier topic.

Add States and Transitions

5. Use the  button to add five states on your state diagram. The states are added with default names $s0$, $s1$, $s2$, $s3$ and $s4$.

Notice that the first state you add is assumed to be the start state and is drawn in green with a double outline. The other states are drawn in cyan with a single outline.

6. Use the  button to add transitions between the states as shown in the picture below.




Notice that when you add more than one transition leaving a state, the transition priority is indicated by a number associated with the transition arc. The priorities are initially assigned in the order that you add the transitions but will be re-assigned in a later topic if necessary.






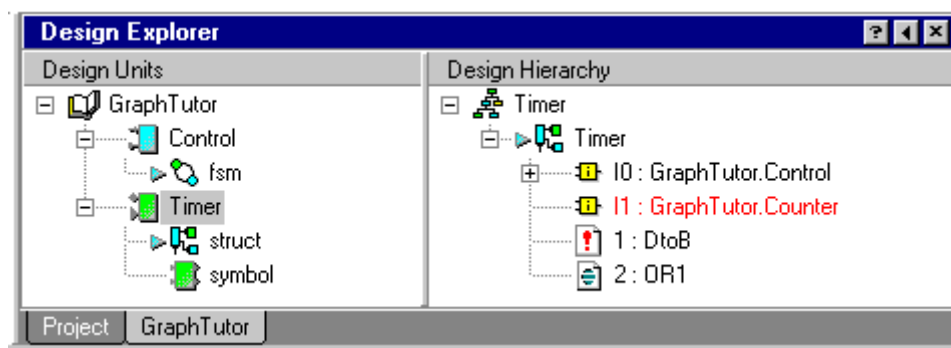
If you add a transition in the wrong direction, you can easily change its direction by choosing **Reverse Direction** from the popup or **Diagram** menu. Note that a popup description (known as an object tip) is displayed when the cursor is stationary over an object. In particular, when the cursor is over a transition, the source state and the destination state are named even if the states are outside the current window.



Save the State Diagram

- Use the  button to save the state diagram. The state diagram was created as a child view from its parent block diagram and is saved using the library, design unit and view names specified when it was created. The design explorer is updated to display the *Control* design unit.

The *Control* design unit is shown as a block in the design explorer because its interface is defined by the connections on its parent block diagram. The *Timer* design unit is shown as a component because it has no parent block diagram and its interface is defined by a symbol.

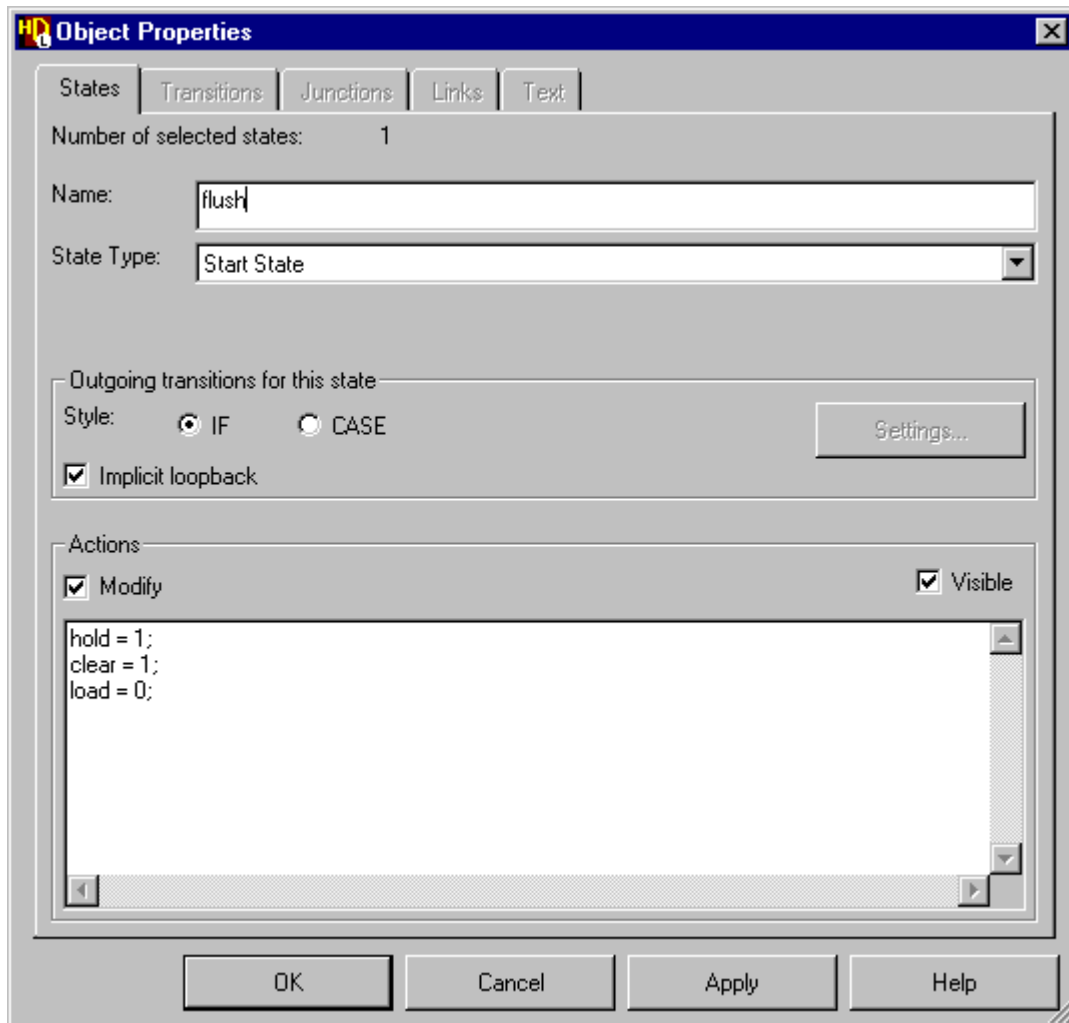
- Click on the  icon for the *Control* design unit in the design explorer to expand the design unit revealing that it contains a state diagram view .
- Select the *Timer* design unit and choose **Show Hierarchy** from the popup menu.
- Select the  Timer hierarchy node in the *Design Hierarchy* pane and choose **Expand All** from the popup menu to reveal the full hierarchy:



Notice that instance *I0* in the hierarchy for the *Timer(struct)* view contains  and  icons showing that the state diagram is described by a concurrent state machine view. The *ORI* embedded block is shown as a text view but the *DtoB* embedded block and the *Counter* instance are not yet defined.

Edit the States

11. Select the start state (*s0* in the picture on [page 2-26](#)) and use the  button to display the **States** tab of the State Machine Object Properties dialog box:



You can also display the dialog box by choosing **Object Properties** from the **Edit** menu or by double-clicking on a state.

The **States** tab allows you to enter a name and actions text for one or more selected states on a state diagram. You can also change the visibility of state actions and (when a single state is selected) change the state to a start state or a hierarchical state.

12. Use the dialog box to enter the following state names and actions replacing the default state names *s0* to *s4* in the picture on [page 2-26](#):

Old Name	New Name	Style	Actions
s0	flush	IF	hold = 1; clear = 1; load = 0;
s1	count	IF	(no actions)
s2	getkey	IF	hold = 1; clear = 0; load = 0;
s3	load_t	IF	hold = 1; clear = 0; load = 0;
s4	load_u	IF	hold = 1 ; clear = 0; load = 1 ;

The Verilog Expression Builder dialog box is automatically displayed when you start to enter the actions. The dialog box can be used to choose from lists of the available port or local signal names, operators and example values. For example, choose the *hold* signal, button, value '1' and button to enter the action *hold = 1;*.



The Expression Builder dialog box can be explicitly displayed at any time by choosing **Expression Builder** from the **Edit** menu.

The syntax for state actions is automatically checked and any errors reported on entry. Verilog statements must be terminated by a semicolon (;) character. Line breaks and spaces can be used for clarity and will be preserved on the diagram.



You can also edit the state name or actions by direct text editing on the diagram or copy a state and paste its state actions into another state by choosing the **Paste State Actions** option from the **Paste Special** cascade in the popup menu.

If the name is larger than the state, it auto-resizes to fit the new name. You can also resize a state by selecting the state and dragging one of its resize handles but you cannot make it smaller than the enclosed name.



Edit the Transitions




13. Add conditions to the transitions as shown in the following table:

Origin	Destination	Priority	Condition
count	flush	1	stop
getkey	flush	3	stop
getkey	count	1	start
getkey	load_u	2	d!=NOUGHTS
flush	load_u	1	d!=NOUGHTS
load_u	load_t	1	(none)
load_t	getkey	1	d!=NOUGHTS





The *NOUGHTS* text string is the name of a constant which will be declared as a state machine property later in this tutorial.

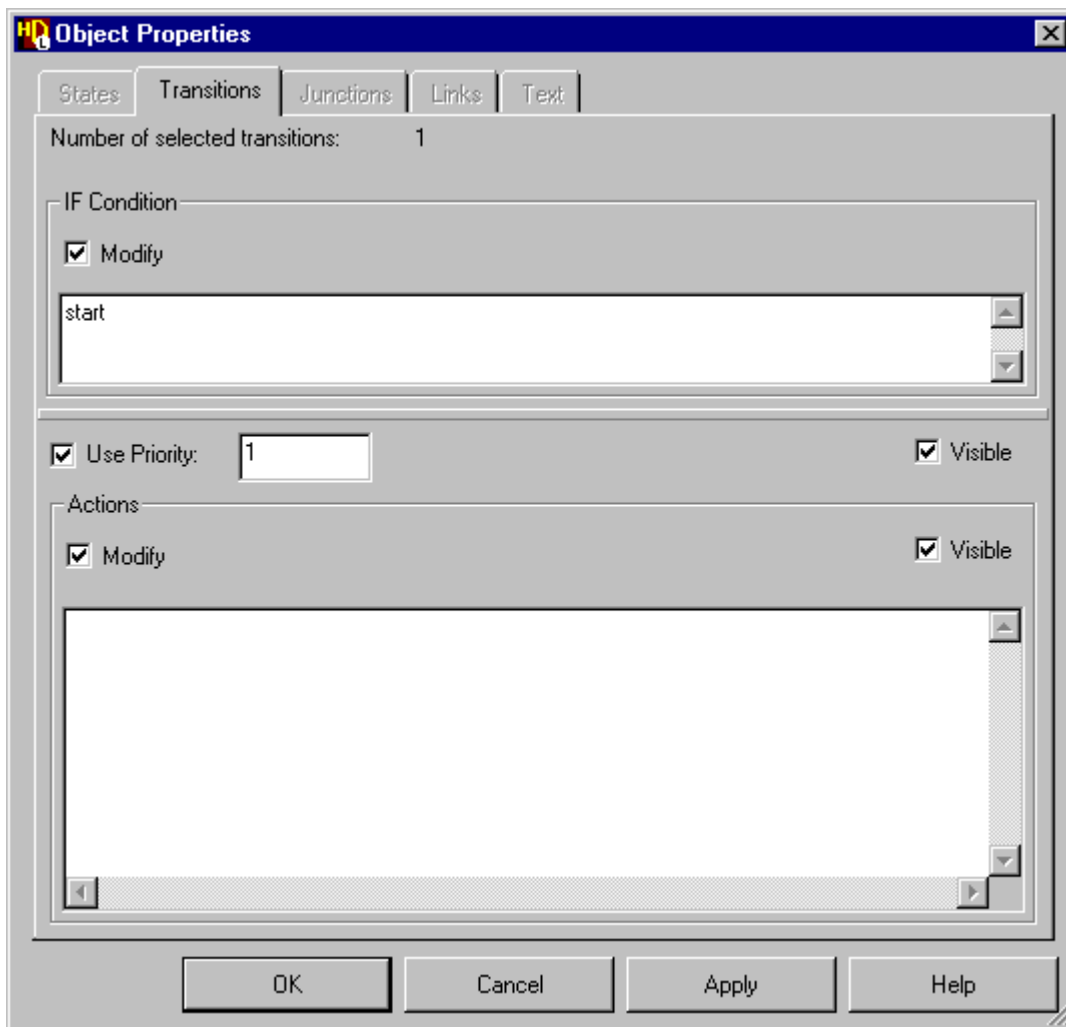
Hold down the  mouse button and select the two transitions entering the state **flush** by dragging the cursor across them (or  mouse button).

Use the  button to display the **Transitions** tab of the State Machine Object Properties dialog box. Enter the condition text *stop* in the dialog box using the expression builder and click the  button to add this condition to both of the selected transitions. A confirmation dialog box is displayed warning that the transition from state *getkey* to state *count* has no condition and is not the lowest priority. Click  to acknowledge the warning. Repeat this procedure for the *start* and *d!=NOUGHTS* conditions.


The condition syntax is checked on entry. Any valid Verilog fragment can be entered using line breaks and indentation to improve the legibility on the diagram if required.

Ensure that the transition priorities leaving the state *getkey* are the same as those shown in the table. You can change a transition priority in the **Transitions** tab of the State Machine Object Properties dialog box. Alternatively, you can use the  button to zoom in or the  button to view an area and use direct text editing to change a transition priority.

When you change a transition priority, the priority of the other transitions leaving the same state are automatically adjusted.

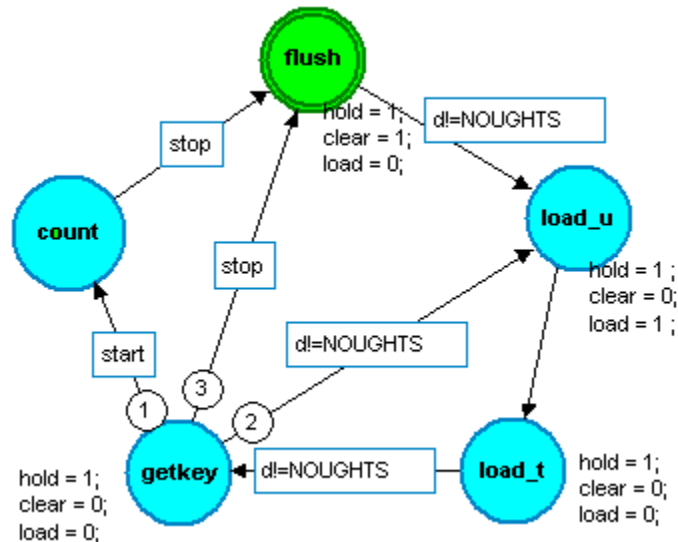


If you have zoomed in, use the  button to view all of the state diagram.

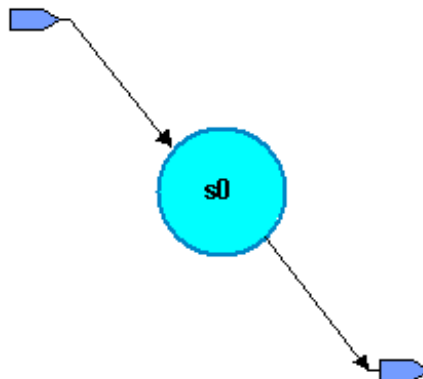
14. Use the  button to save your changes to the state diagram.




Create a Hierarchical State Diagram

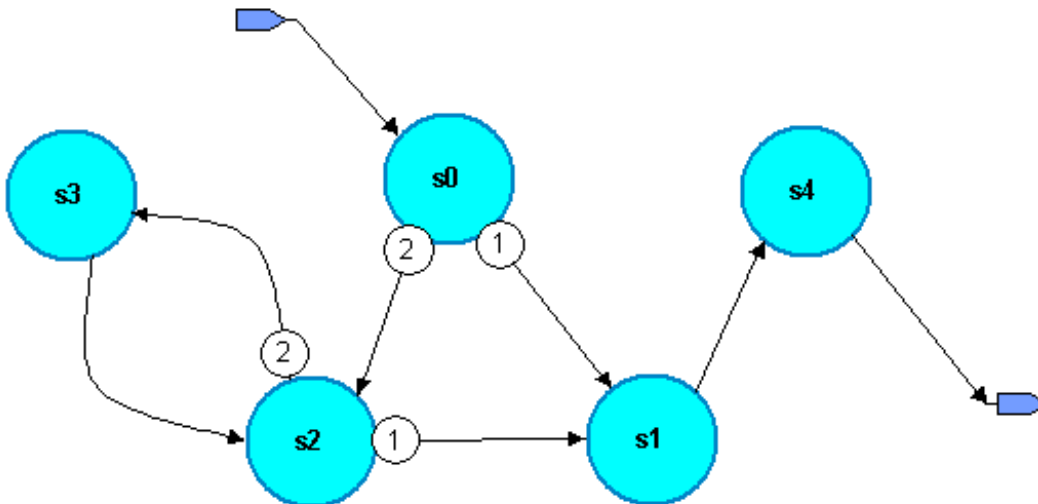
The state diagram should look similar to the following picture.



15. Select the *count* state and choose **Hierarchical State** in the **Change To** cascade from the **Diagram** menu. Alternatively, choose the Hierarchical State pulldown option for the State Type in the **States** tab of the State Machine Object Properties dialog box. The *count* state is redrawn as a hierarchical state with a triple outline and darker fill color.
16. Double-click on the hierarchical state (or use the **Open Down** option from the popup menu) to create the new hierarchical child state diagram. A new child state diagram is opened as a tabbed window initialized with a default state *s0* connected to an entry point and exit point.



17. Notice that the name of the hierarchical state *count* is included in the window title: *GraphTutor/Control/fsm [csm/count]*. This convention shows that the child diagram is a partial view of the parent diagram.
18. Select the exit point and choose **State** in the **Change to cascade** of the **Diagram** menu.
19. While the new state is selected, drag with the  mouse button and release the mouse button with the ghosted state to the left and below the first state. Use the **Copy Here** option from the popup menu to make a copy of the state at the cursor position.
20. Repeat this procedure to add two more states on the diagram. This method for adding objects which can be useful when you want to add an object with the same or similar properties and attributes to an existing object.
21. Use the  button to add a new exit point and the  button to connect transitions between the states as shown in the following picture:



You can add route points by clicking at several points between states to create a smooth arc as shown in the picture between states *s2* and *s3*. You are not limited by the page boundaries when you add objects or edit a graphical diagram. If necessary, you can drag a rectangle around the objects to select them and move them back inside the page boundary or choose **Refresh Page Boundaries** from the **File** menu to show how the edited diagram will be fitted onto printer pages.

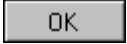
Complete the Hierarchical State Diagram


22. Use the **States** tab of the State Machine Object Properties dialog box to rename the states and add state actions as shown below:

Old Name	New Name	Style	Actions
s0	standby	IF	hold = 1; clear = 0; load = 0;
s1	alarm	IF	hold = 1; clear = 1; load = 0; beep = 1;
s2	counting	IF	hold = 0; clear = 0; load = 0;
s3	suspended	IF	hold = 1; clear = 0; load = 0;
s4	end_count	IF	hold = 1; clear = 1; load = 0;

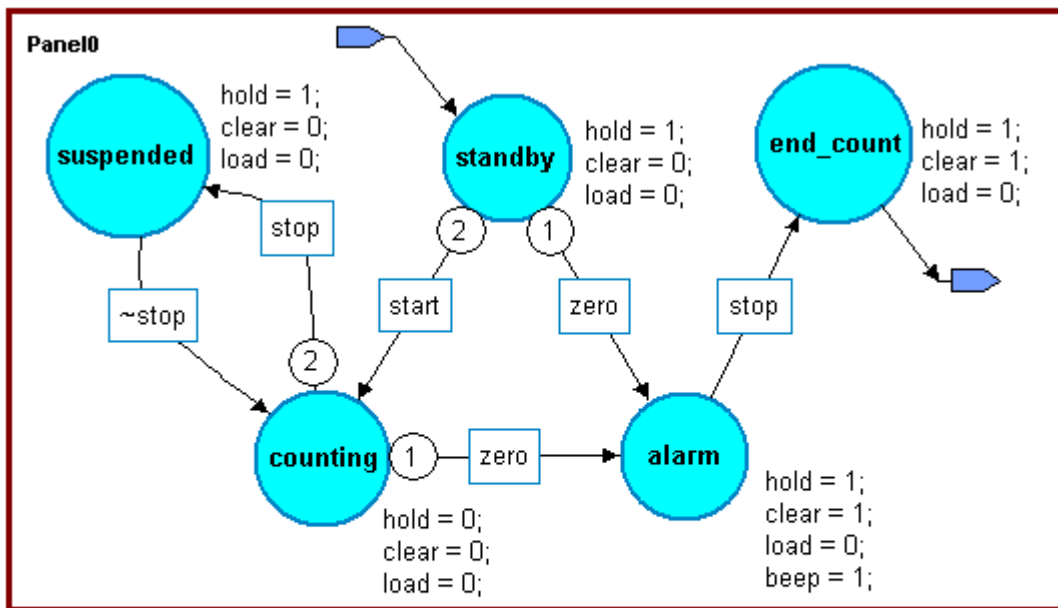
23. Use the **Transitions** tab to add the following conditions:

Origin State	Destination State	Priority	Condition	Actions
suspended	counting	1	~stop	none
counting	suspended	2	stop	none
counting	alarm	1	zero	none
standby	counting	2	start	none
standby	alarm	1	zero	none
alarm	end_count	1	stop	none


A confirmation dialog box may be displayed warning that a transition you have not yet edited has no condition and is not the lowest priority. Click the  button to acknowledge the warning.


Complete the hierarchical state diagram by editing the title and comments in the title block and using the  button to add a panel around the graphical objects on your diagram. This panel can be used to set the diagram view when you animate the state diagram later in this tutorial.

The state diagram should look similar to the picture below:



Mentor Graphics		Project:	Tutorials
Title:	Counter state machine	Child hierarchical view of the count state in the Counter state machine.	
Path:	GraphTutor/Control/fsm		
Edited:	by joans on 04 Apr 2003		



 Ensure that the transition priorities are as shown and that you include the terminating semi-colon when you enter the state actions.

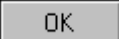
24. Use the  button to save the state diagram.

Although it is displayed in a separate tab, a child hierarchical diagram is a partial view of a state machine and the parent and child diagrams are saved as a single design unit view (*GraphTutor/Control/fsm.sm*).

Editing State Machine Properties

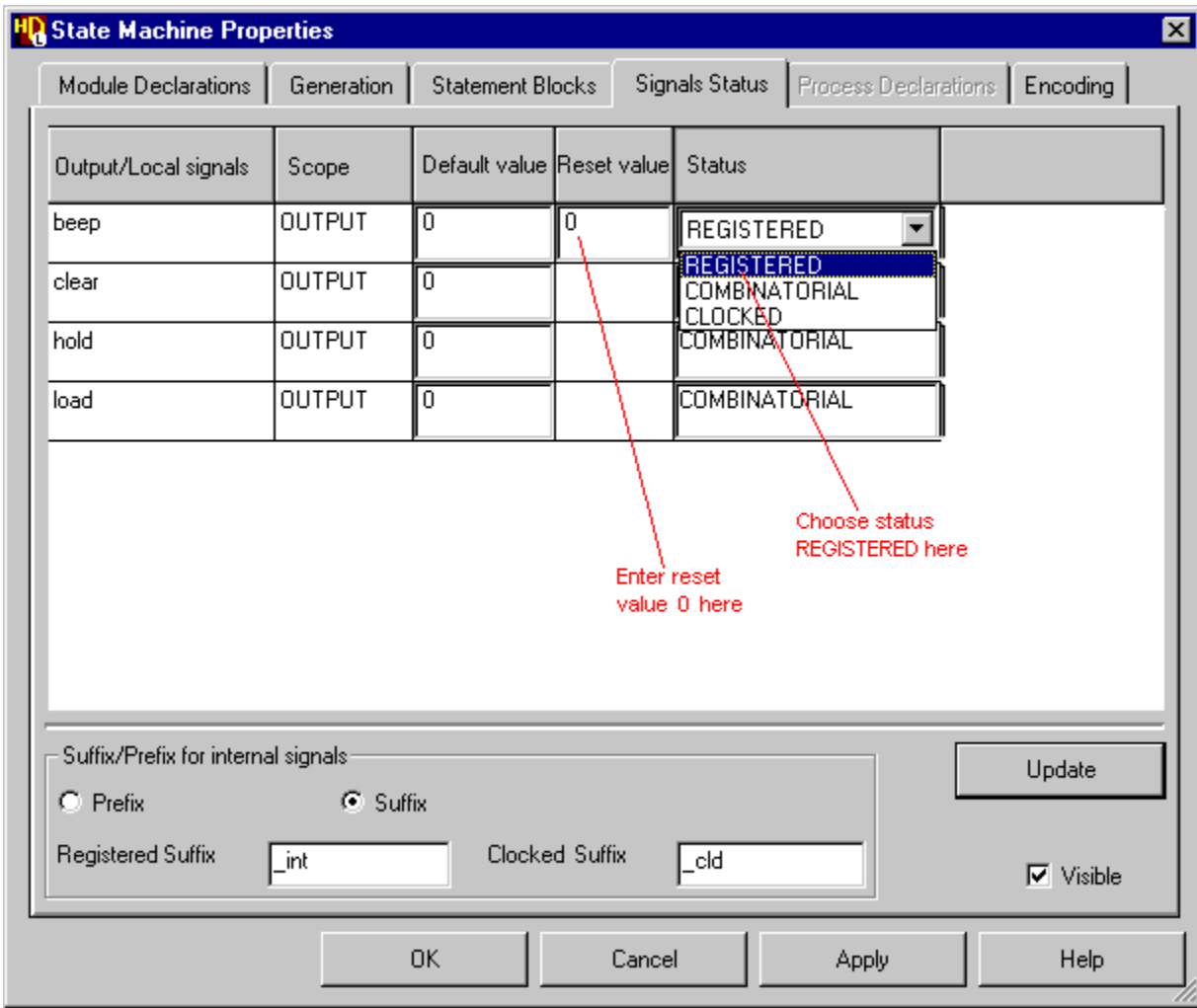
There are a number of properties associated with a state machine which can be edited using the State Machine Properties dialog box. The dialog box can be accessed by choosing **State Machine Properties** from the **Diagram** or popup menu in the parent or child state diagram. It can also be accessed by double-clicking over one of the labels which are displayed in the parent diagram of a hierarchical state machine.

25. Use the  button (or choose the **Open Up** option from the **File** menu) to re-display the parent state diagram if it is not already displayed.
26. Double-click the  mouse button over the Global Actions or Concurrent Statements or State Registers label to display the **Statement Blocks** tab of the State Machine Properties dialog box. There are no global actions, concurrent statements or state register statements required in this design. Hide the labels for these object on the state diagram by clearing the **Visible** check box for Global Actions, Concurrent Statements and State Register Statements in the **Statement Blocks** tab.
27. Choose the **Module Declarations** tab and enter the following declaration for the 10-bit constant NOUGHTS in the free-format entry box:

```
parameter NOUGHTS = 10'b0;
```
28. Click the  button to confirm the dialog box and update the diagram. The constant declaration is added below the Module Declarations label on the state diagram and will be included as module declarations when HDL is generated for the state machine. All other labels other than the default signals status should now be hidden on the diagram.
29. Re-display the State Machine Properties dialog box and choose the **Signals Status** tab by double-clicking over the Signals Status label on the diagram.

Notice how the output signals are listed in the dialog box with the default value set to 0 and the status *COMBINATORIAL*.
30. Click on the Status field for the *beep* signal and choose *REGISTERED* from the drop down box.


31. Click in the Reset value field and enter the value 0.



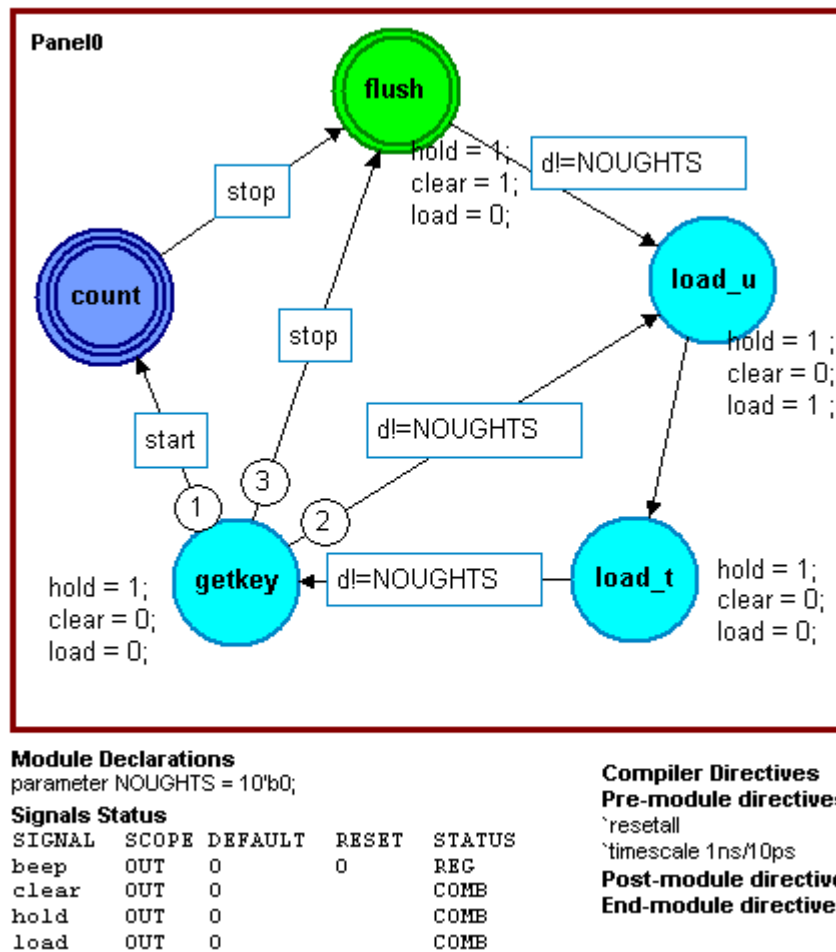
Any locally declared internal signals would be shown with the status REGISTERED. Bidirectional or buffer signals are treated as output signals.

The dialog box also allows you to change the suffix or prefix used for internal registered or clocked signal names. For this tutorial, suffixes are used with the default values *_int* and *_cld*.


32. Confirm the dialog box by clicking the  button.

33. Complete the state diagram by editing the title and comments in the title block and using the  button to add a panel around the graphical objects on your diagram. This panel can be used to set the diagram view when you animate the state diagram later in this tutorial.

The final state diagram should look similar to the picture below:

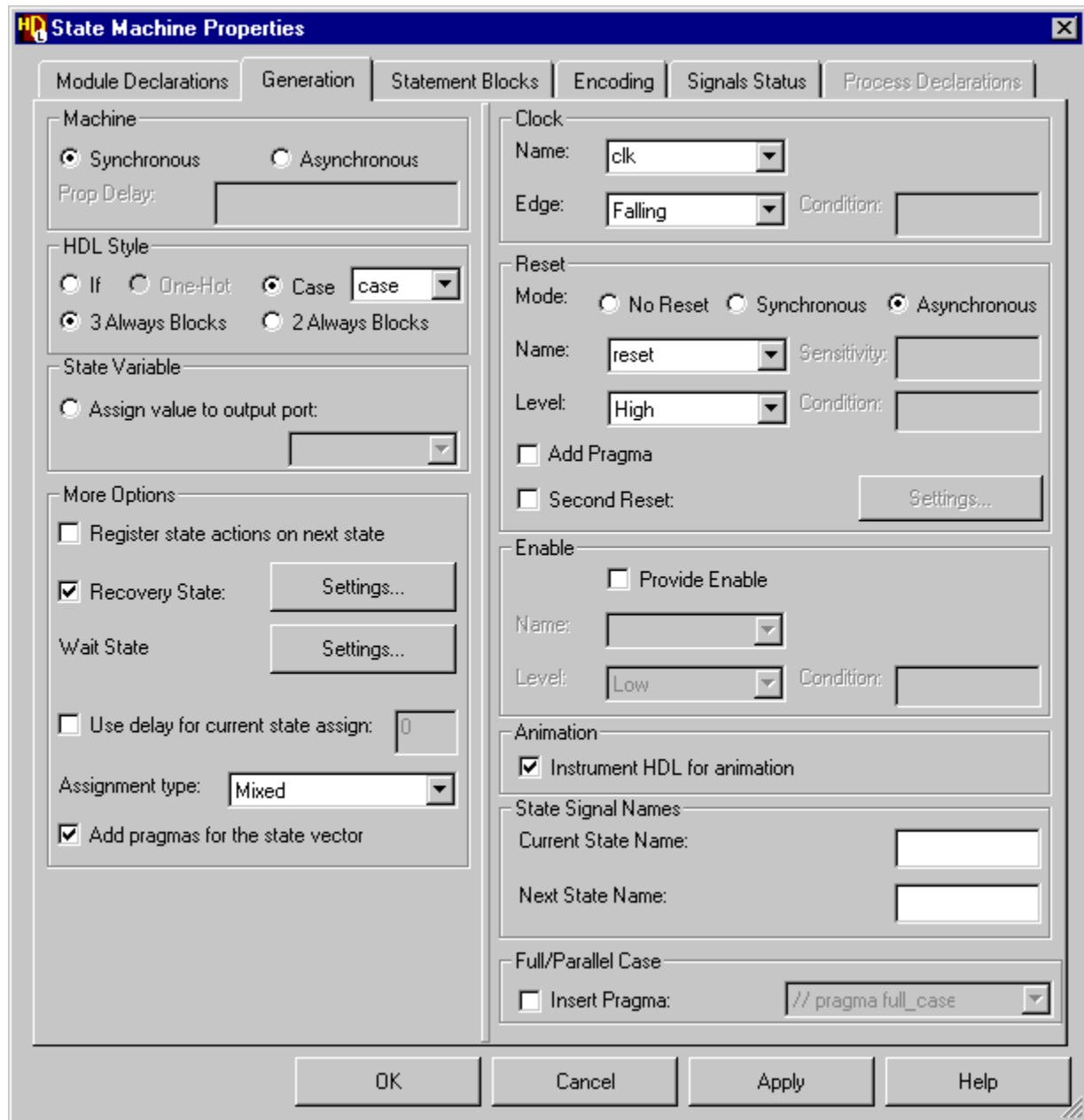


Notice that the *beep* signal (in the *alarm* state on the child hierarchical state diagram) has been replaced by the internal signal name *beep_int*.


The State machine Properties dialog box also provides tabs for setting HDL generation characteristics and state machine encoding. State machine encoding is not used in this tutorial and the encoding mode can be left as **Hard Automatic**. You can use the  buttons for more information about each tab of the State Machine Properties dialog box.

Set Generation Properties

34. Re-display the State Machine Properties dialog box if necessary and choose the **Generation** tab. This tab sets properties used for HDL generation.



35. Choose the **Synchronous** option in the Machine box. Use the default options **Case** and **3 Always Blocks** for HDL Style and leave the **State Variable** option unassigned.





36. Check that the **Recovery State** is set to `<start_state>` by using the  button to display the Recovery State Settings dialog box.



This entry automatically assigns the start state or you can choose from a pulldown list of states and specify recovery state actions.

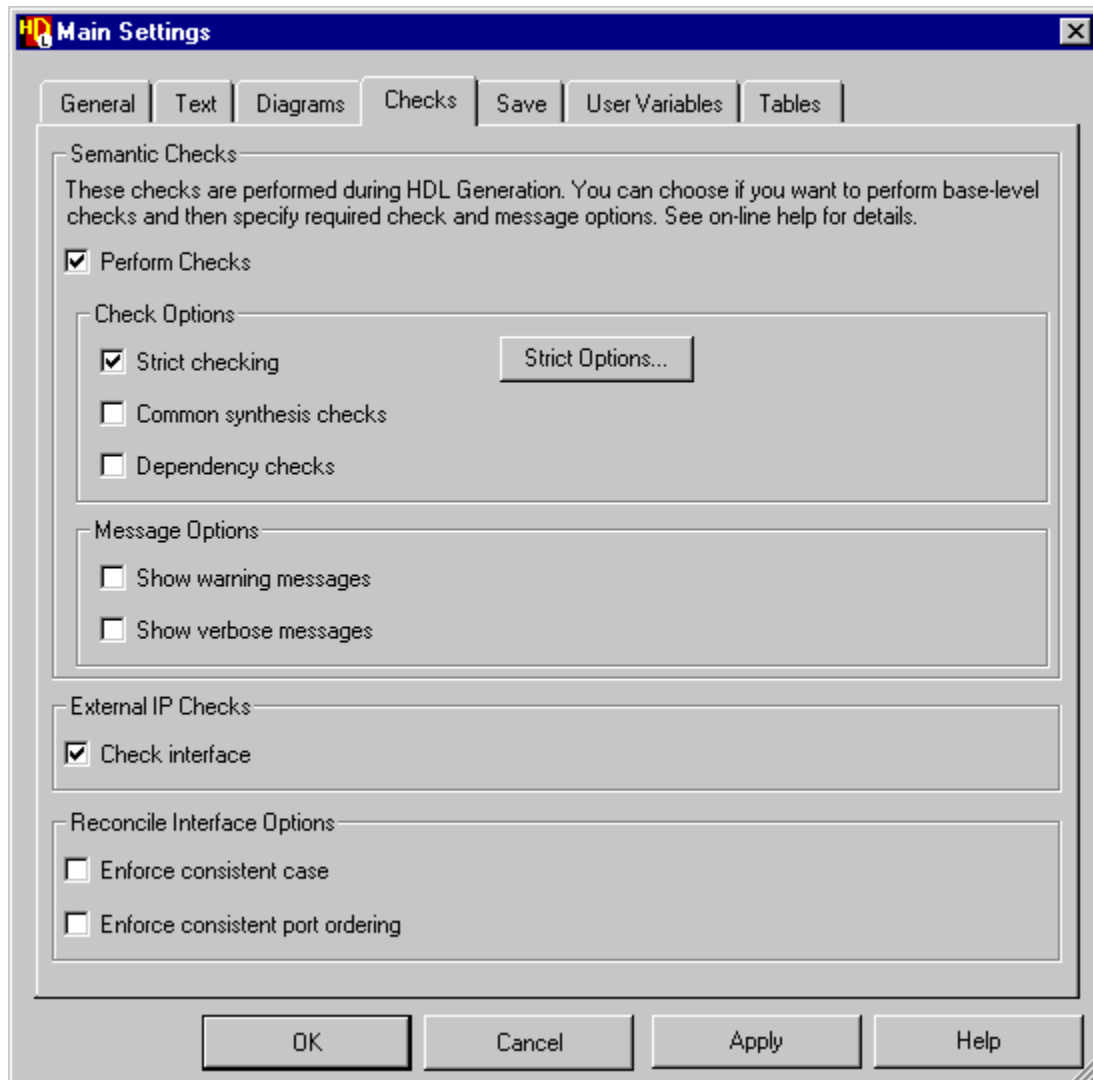
37. Leave the **Register state actions on next state** and **Use delay for current state assign** options unset, and leave the **Assignment type** set to Mixed.
38. Choose *clk* from the pulldown list of clock signal names and *Falling* from the Edge pulldown list. Set an **Asynchronous** reset and choose the signal name *reset* with a *High* level from the pulldown lists. Leave the **Provide Enable** options unset.
39. If a ModelSim or NC-Sim simulator is available, set the **Instrument HDL for animation** option. This option adds additional code to the generated HDL which is used for state machine animation later in this tutorial.

 This additional code is surrounded by translation control pragmas which ensure that it is ignored by downstream synthesis tools.

40. Use the  button (or  followed by ) to apply these generation characteristics to your state machine and dismiss the dialog box.
41. Use the  button to save the state diagram.

Set Checks for HDL Generation

You can set the level of checks performed when HDL is generated by using the **Checks** tab in the Main Settings dialog box which can be accessed by choosing **Main** from the **Options** menu.




For this tutorial, the **Strict checking** option should be selected.



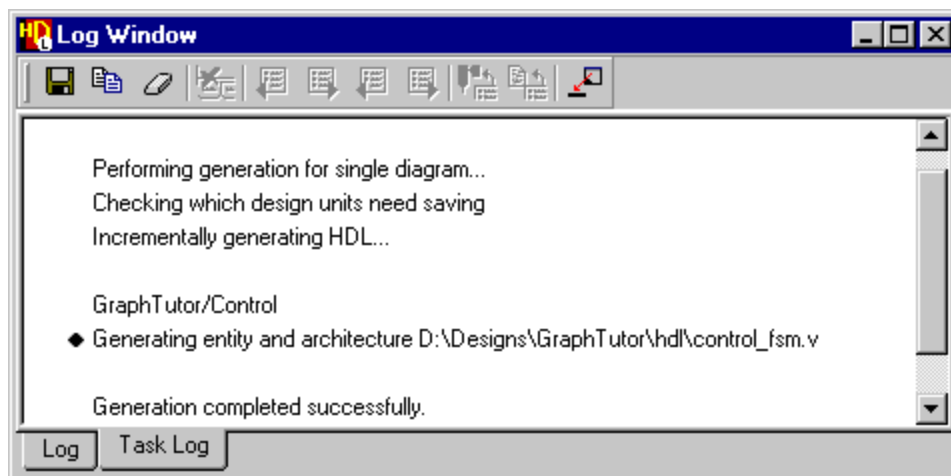
The **Strict Options...** button can be used to display a dialog box which allows you to modify the strict checking options. The default strict check options should be used for this tutorial.





Generate HDL for the State Machine



Although the *Timer* design is not yet complete, you can now generate HDL for the *Control* state machine.


42. Choose the **Run Single** option from the **Generate** cascade in the **Tasks** menu (or use the  button) in the state diagram window.

The progress of the HDL generation is monitored in a log window which includes any errors and warnings issued during generation.





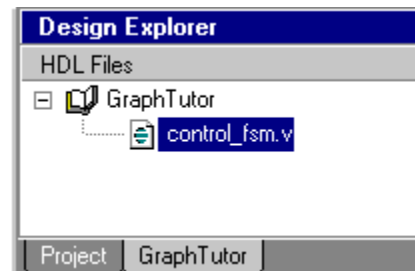
If there are any errors, you can move to the next error message using the  button or the previous error using the  button. You can display the source graphics corresponding to the error by double-clicking on the error message (or using the  button when the error is selected). Alternatively, you can use the  button when the error is selected to view the generated HDL. If your text editor supports a line number argument, the HDL line corresponding to the error is selected automatically.

If there are no errors, you can use the toolbar buttons  and  to view the source graphics or generated HDL corresponding to any message which is marked by the ◆ icon in the log window. For example, the “Generating module ...” message shown above.


Generated HDL files can also be opened by using the  button in any graphics editor window or when the diagram view is selected in the design explorer.

Display HDL Files in the Design Explorer

43. Use the  button (or choose HDL Files from the **Mode** cascade in the **View** menu) to display the *HDL Files* view in the design explorer.
44. Click on the  icon for the *GraphTutor* library and notice that the view is expanded to reveal the generated Verilog module file for the *Control* design unit



45. Open the HDL file by double-clicking on the generated file name or by choosing **Open File** from the popup menu.



If you are using the DesignPad HDL text editor, you can use the  button or the **Trace To Graphics** command from the **Document** menu in DesignPad to cross-reference from a line in the generated HDL to the corresponding graphics object.

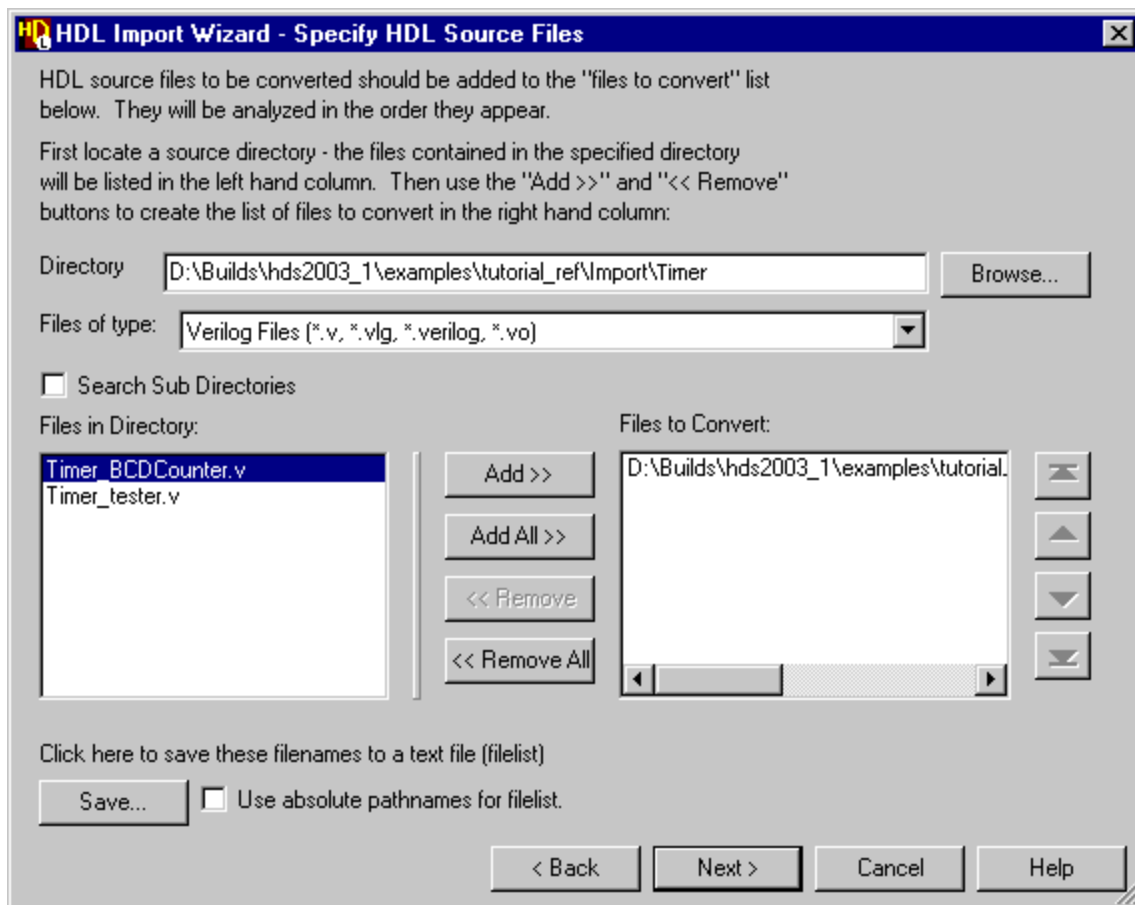
A similar option can be set up for other user customizable external text editors. Refer to the “Text Editors” chapter in the *HDL Designer Series User Manual* for more information.

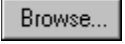
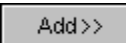
46. Close the text editor after you have viewed the generated HDL.
47. Use the **Close Window** option from the **File** menu to close the parent *GraphTutor/Control/fsm [csm]* and the child *GraphTutor/Control/fsm [csm/count]* views of the state machine.



Import the BCDCounter Design Unit

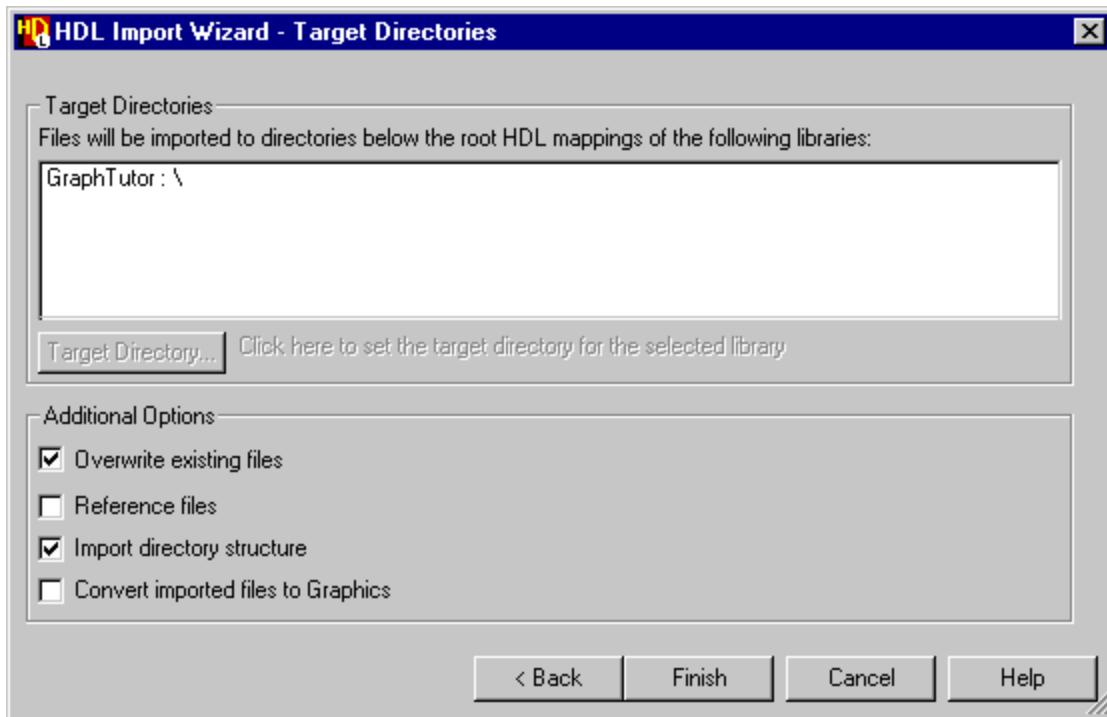
You have now defined the *Control* block by drawing a graphical state machine. The *Counter* block will be defined by instantiating two instances of a HDL text component in a child block diagram.


1. Use the  button (or choose **HDL Import** from the **HDL** menu) to display the HDL Import wizard.
2. Choose **Specify HDL files** and **Verilog** in the first page of the wizard and click the  button to display the Specify HDL Source Files page.



3. Choose Verilog files from the pulldown filter for Files of type and use the  button to locate the `..\examples\tutorial_ref\Import\Timer` installation subdirectory. Select the `Timer_BCDCounter.v` file and add it to the list of files for conversion by using the  button.

4. Click the  button on the wizard to display the Target Libraries page. This page allows you to select the default target library and add libraries if there are any instantiations for black box components in the source HDL code.
5. Ensure that your *GraphTutor* library is selected as the target library and click the  button to display the Target Directories page. This page allows you to control the imported directory structure and set additional HDL import options:



6. Click the  button in this page to complete the HDL import using the default target directory and options specified in the wizard. The following completion message should be displayed in the HDL log window:

```
** Importing 'D:\Designs\GraphTutor\hdl\Timer_BCDCounter.v'  
HDL Import complete
```

```
-----  
  
1 file imported to 1 library  
  
-----
```

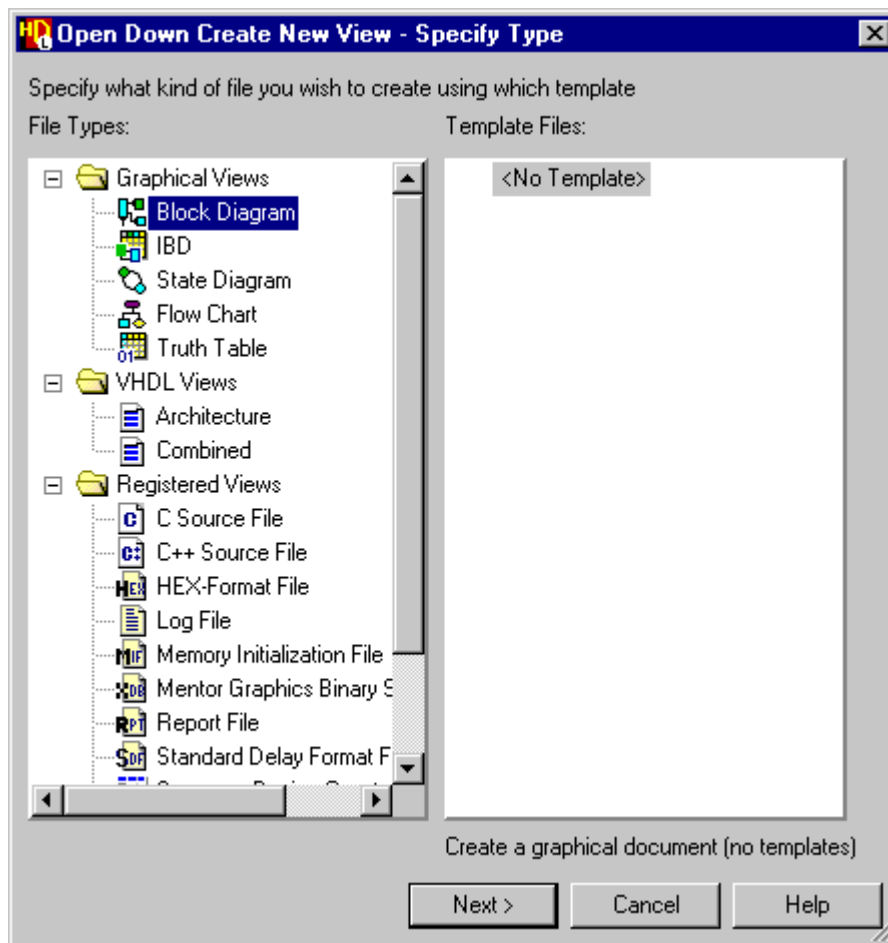
Create a Child Block Diagram


1. Display the *Timer* block diagram.




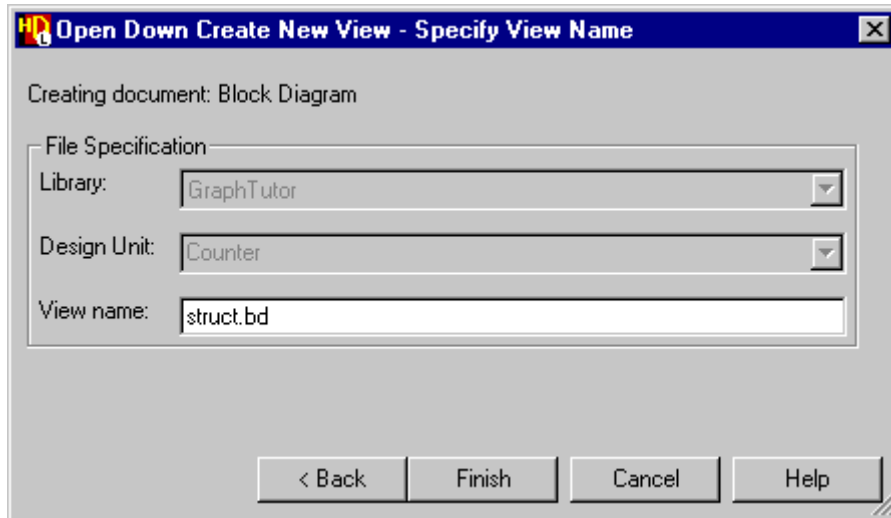
If the window is obscured, you can pop it to the front by selecting the window name from the **Windows** menu list in any other window.

2. Double-click on the body of the *Counter* block (or choose New View from the **Open As** cascade of the popup menu) to display the Open Down Create New View dialog box.



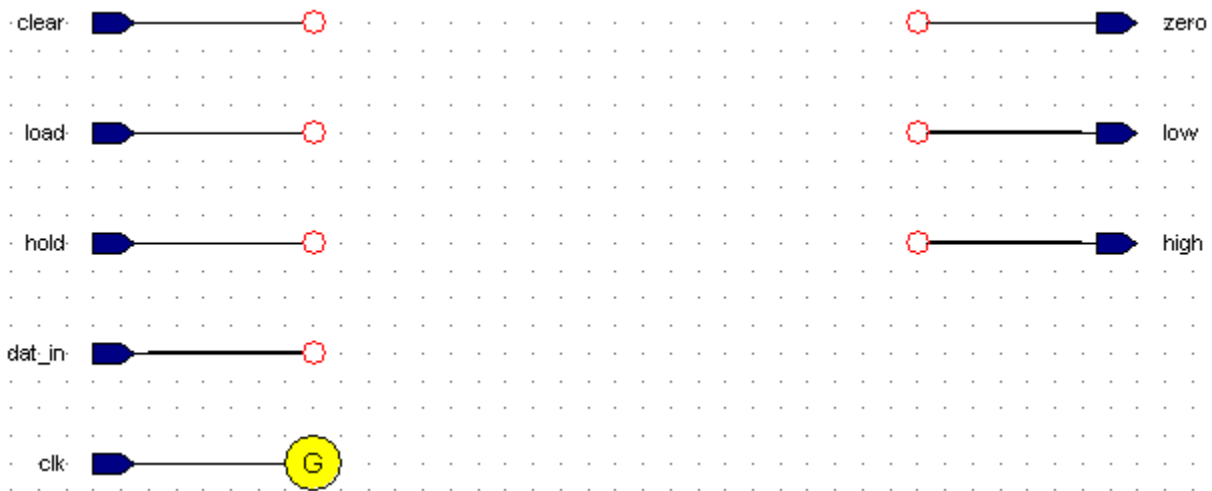
3. Select *Block Diagram* from the list of file types and use the  button to display the Specify View Name page of the wizard.

4. Use the  button to confirm the wizard with the default view name *struct.bd*.



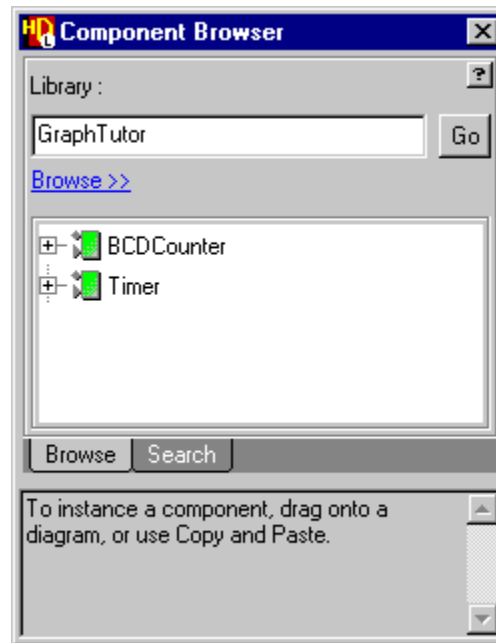
A new block diagram *GraphTutor/Counter/struct* is created as a child view of the *Counter* block. The block diagram is initialized with declarations and ports corresponding to the interface signals and buses on the parent diagram (including a global connector for the *clk* signal).

Notice how inputs are initialized on the left and outputs on the right.

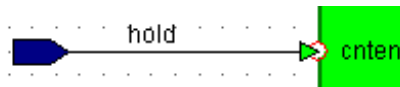



5. Use the  button (or choose **Component** from the **Add** menu) to display the Component Browser.



- Click on [Browse >>](#) to display a drop down list of libraries and double-click to choose your *GraphTutor* library:

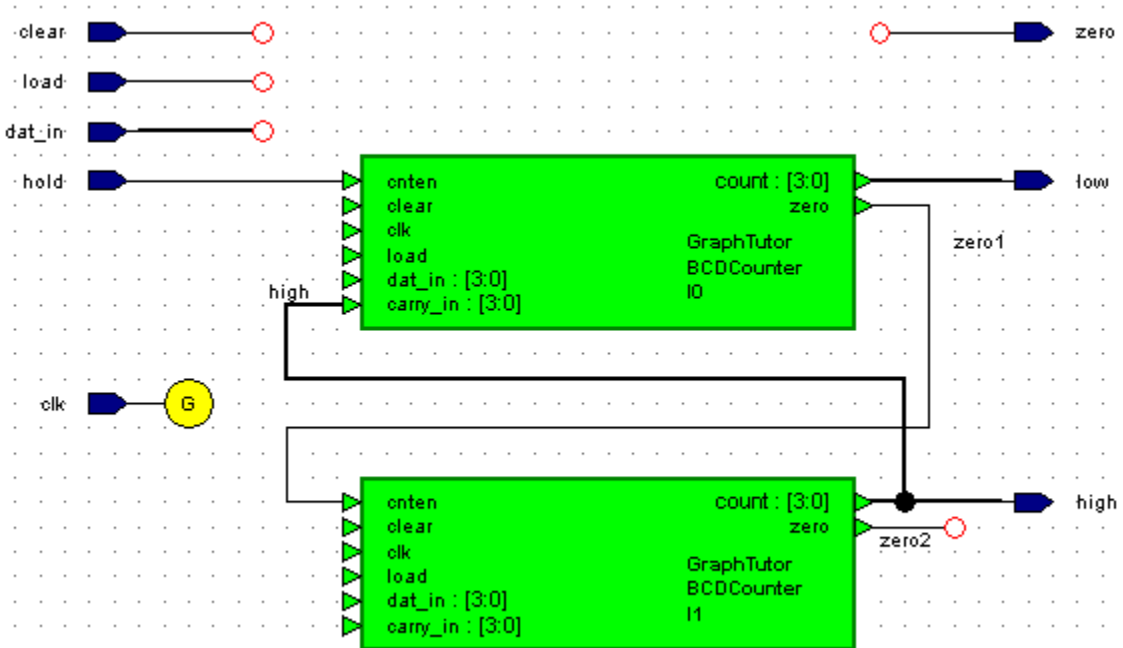



- Drag two instantiations (*I0* and *I1*) of the *BCDCounter* component from the Component Browser placing one below the other on the block diagram but allowing several grid lines for connections between them.
- Move the *hold* signal so that it overlaps the *cnten* port on instance *I0*.





- Similarly, move the *low* signal over the *count* output port on instance *I0* and the *high* signal over the *count* output port on instance *I1*.
- Select both instances and choose **Connect** from the popup menu to make the connections.
- Use the  button to connect a signal between the *zero* output port on *I0* and the *cnten* port on *I1*. This signal is automatically named *zero1* since the output signal *zero* already exists on the diagram.

12. Use the  button to connect a bus between a point on the *high* output bus and the *carry_in* port on *I0*.
13. Use the  button to connect a stub signal to the zero output port on *I1*. This signal is automatically named *zero2* since the signals *zero* and *zero1* already exist on the diagram. The block diagram should now look similar to the following picture:



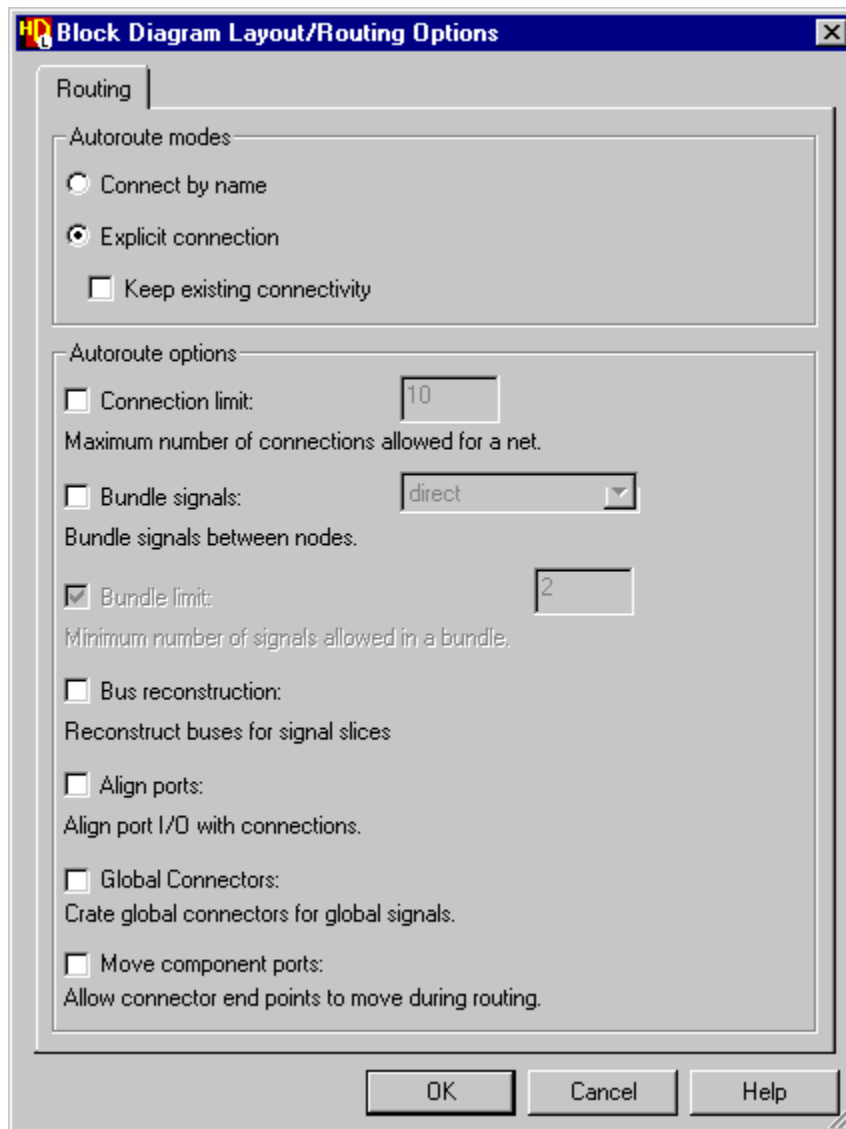
14. Select both instances *I0* and *I1* of the *BCDCounter* and choose **Add Signal Stubs** from the popup menu to display the Add Signal Stubs dialog box.
15. Click the  button to confirm that you want to add signal stubs to all port types. You are warned that the nets *clear*, *clk*, *dat_in* and *load* already exist on the diagram.

 Do not set the **Create unique net names** option. This option could be used when you do not want to automatically connect nets with the same net but is not required in this tutorial.

16. Click the  button to acknowledge the warning. The signal stubs are added when you accept the warning and the matching signals are implicitly connected by name.

It is not necessary to connect the nets on the diagram although making implicit connections can make the diagram clearer. You can automatically connect the matching nets on a diagram by using commands in the **Autoroute** cascade from the **Diagram** or popup menu.

17. Check the options for automatic routing by choosing **Autoroute Options** from the **Autoroute** cascade to display the Block Diagram Layout/Routing Options dialog box.



18. Check that **Explicit connection** is set. Unset **Align ports**, **Global Connectors** and **Move component ports** and confirm the dialog box.

19. Select the *clear*, *load* and *dat_in* nets (or the ports with these names on instances *I0* and *I1*). Choose **Autoconnect** from the **Autoroute** cascade of the popup menu.

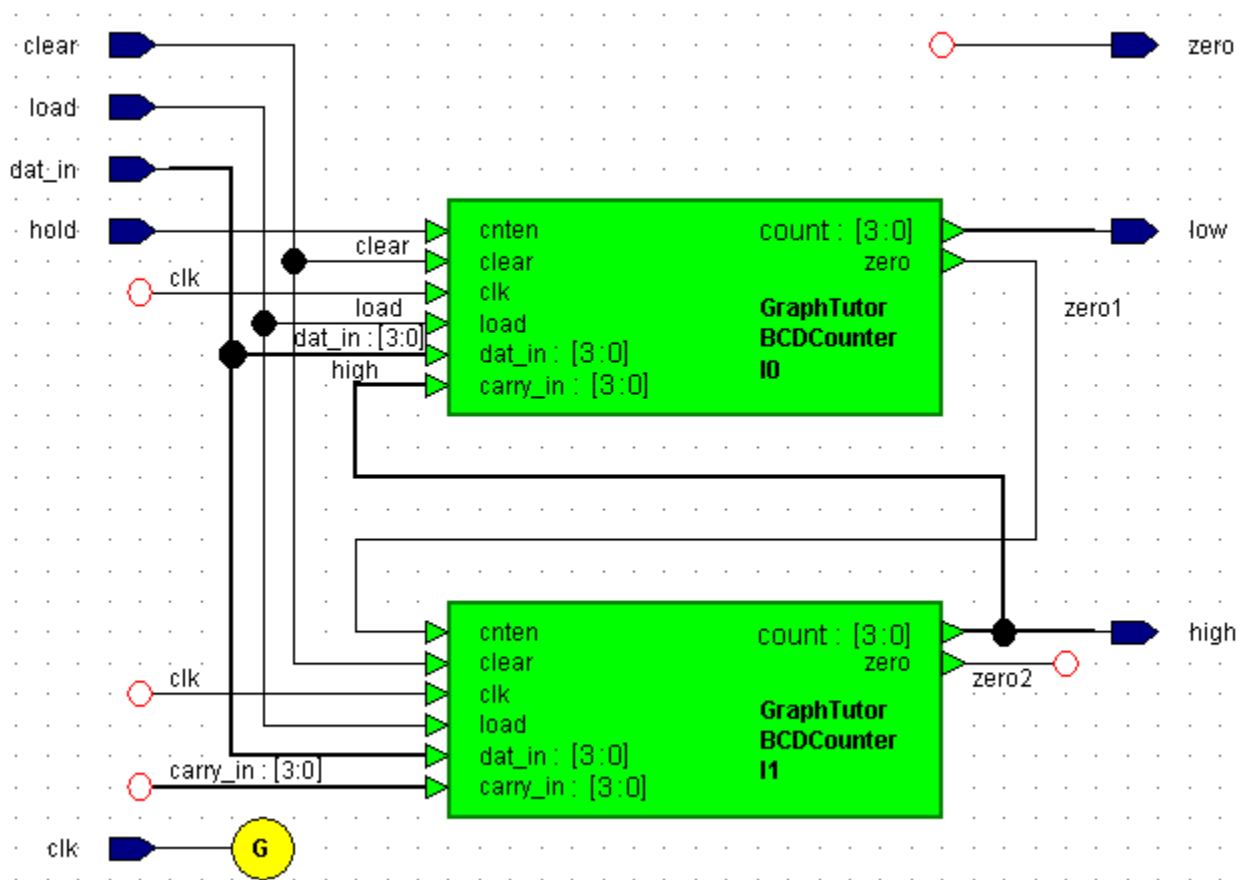
The signals are automatically connected to the matching stub signals on instances *I0* and *I1*.



If nothing is selected on the diagram, autoconnect attempts to re-route all connections on the diagram. Ensure that only the required signals are selected if you do not want to change all the connections on a diagram.


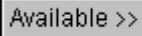

Note that the *clk* port (with its global connector) is connected by name to both *clk* ports on the instances.

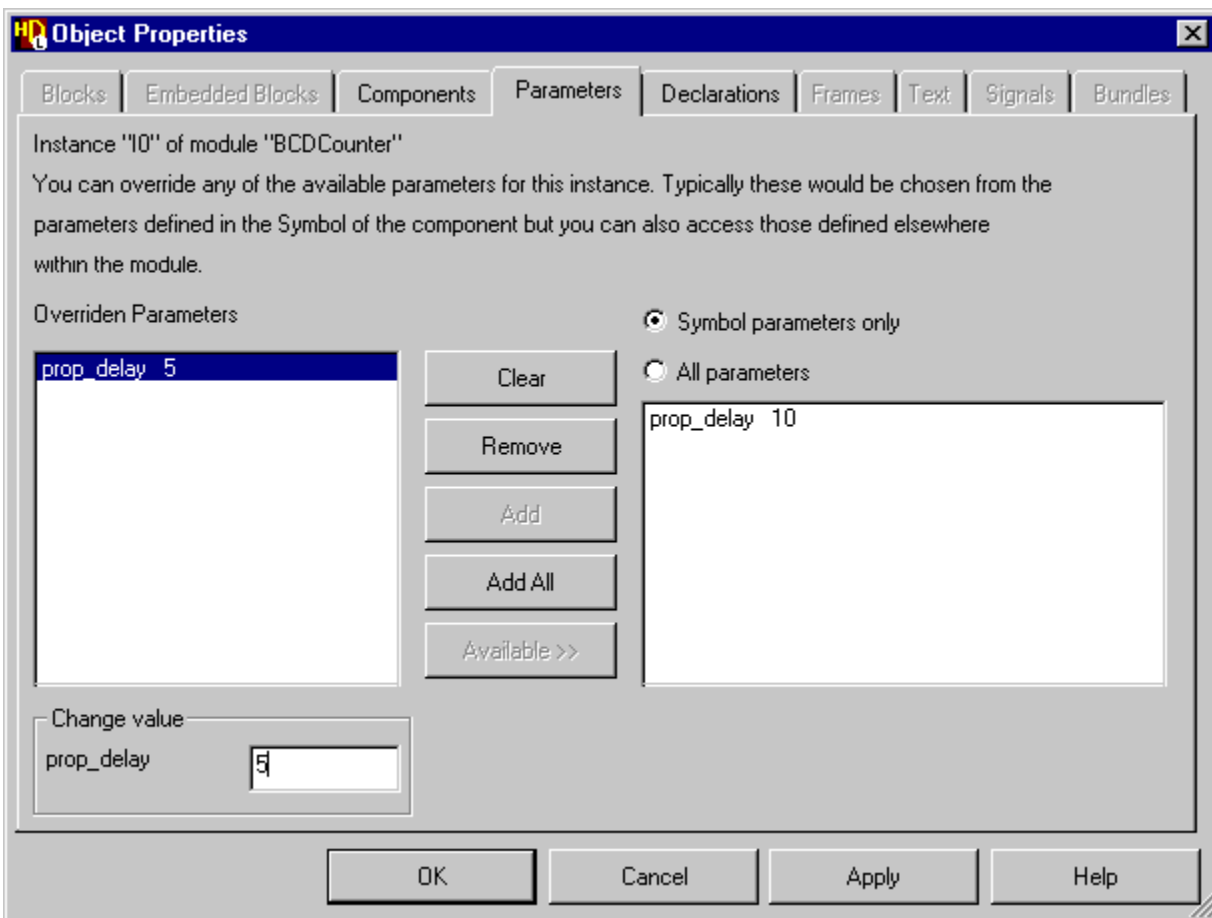
The block diagram should now look similar to the following picture:




Edit the Parameter Mapping

Parameter mapping allows you to modify the value of a Verilog parameter which is declared in the symbol defining the component which can have different values for each instance of the component.

20. Select instance I0 of the *BCDCounter* component, use the  button to display the Block Diagram Object Properties dialog box and choose the **Parameters** tab.
21. Click the  button to display a list of the parameters defined on the symbol and the  button to move the *prop_delay* parameter into the list of overridden parameters.





22. Edit the value in the Change value box to be 5 and use the  button to update the block diagram.

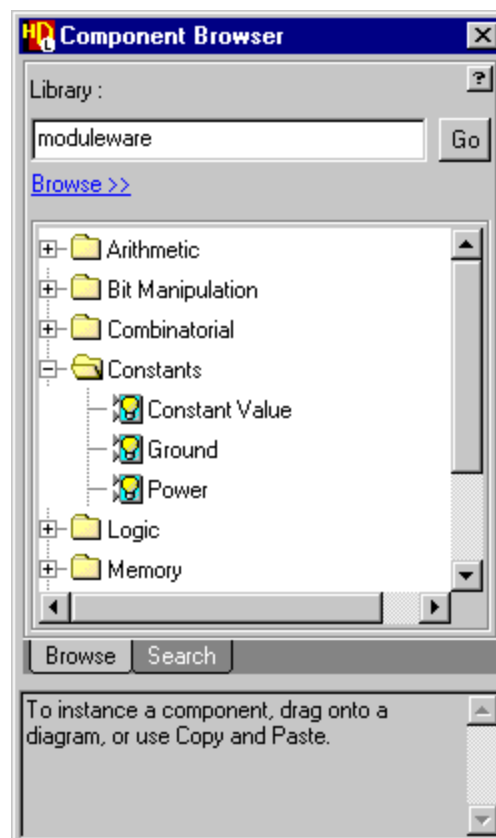
The parameter mapping is added above the component on the diagram as a text object which can be moved independently by dragging it with the **Left** mouse button.


23. Repeat this procedure to set the parameter mapping for the second instance of *BCDCounter*.

Add ModuleWare Components

Although you have routed the *Counter* block diagram, several signals have been left unconnected. These will be connected by using ModuleWare components.

24. Component Browser by using the  button (or by choosing **ModuleWare** from the **Add** menu). The browser displays the contents of the *moduleware* library which is divided into a number of folders.
25. Click on the  icon to expand the folder for the *Constants* category:

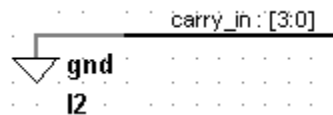


26. Use the  mouse button to drag an instance of the *Ground* component from the Component Browser over the *Counter* block diagram and release the button to place it near the *carry_in* input to instance *I1*.

A *gnd* ModuleWare instance is added with a default instance name (*I2*).

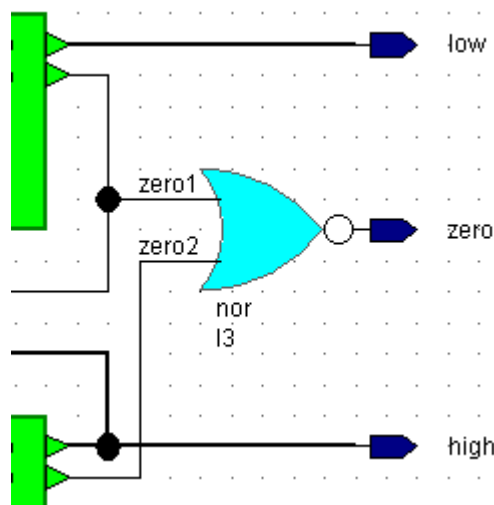
27. Use the mouse to position the stub signal on the *gnd* component over the dangling connector on the *carry_in* net and choose **Connect** from the popup menu.

The *I2* ModuleWare instance should now look similar to the following picture.





Notice that the *carry_in* net is a 4-bit bus. The ModuleWare port width is automatically set to match the width of the connected net.

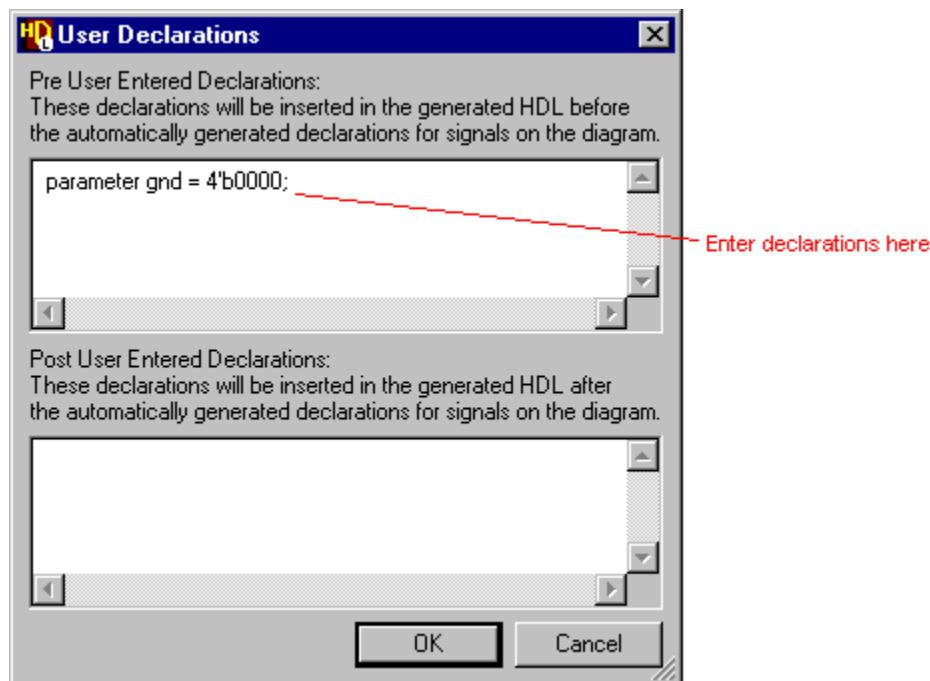
28. Repeat this procedure to add an instance of the *N Input NOR* component from the *Logic* folder of the *moduleware* library near the *zero1* output from instance *I0* on your block diagram.
29. Connect this *nor* instance to the *zero1*, *zero2* and *zero* nets as shown in the following picture:





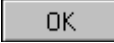
Add a User Declaration

30. Double-click over the Declarations label on the diagram (or use the  button) to display the **Declarations** tab of the block diagram Object Properties dialog box.
31. Click the  button to display a free format entry dialog box. Define the *gnd* signal to be a 4-bit binary zero vector constant by entering the following declaration in the Pre User Entered Declarations section of the dialog box:

```
parameter gnd = 4'b0000;
```



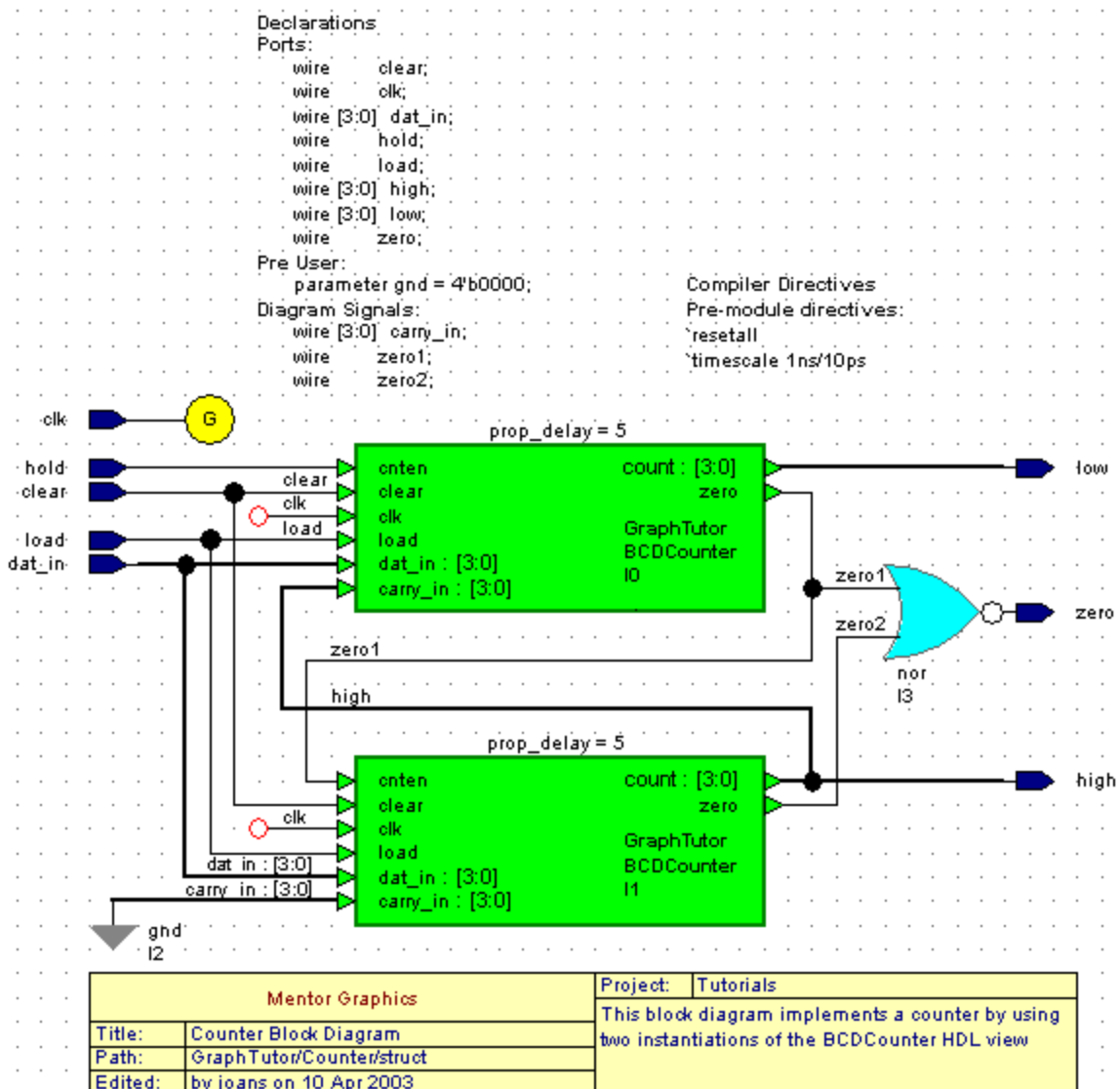
-  The syntax is automatically checked and any errors reported on entry. You can enter any valid Verilog statements which must be terminated by a semicolon (;) character. Line breaks and spaces can be used for clarity and will be preserved on the diagram.


32. Click on the  button to confirm the user declaration and click the  button in the Block Diagram Object Properties dialog box to add the new user declaration on the diagram.

The pre user entered declarations are added to the declarations list on the diagram between the port and signal declarations.

- Complete the block diagram by editing the title block. You may also want to drag objects or groups of objects to re-arrange the diagram within the page boundaries for your default printer.

For example, the following picture shows the *Counter* block diagram re-arranged for portrait orientation.:




- Use the  button to save the block diagram.

Create a Truth Table

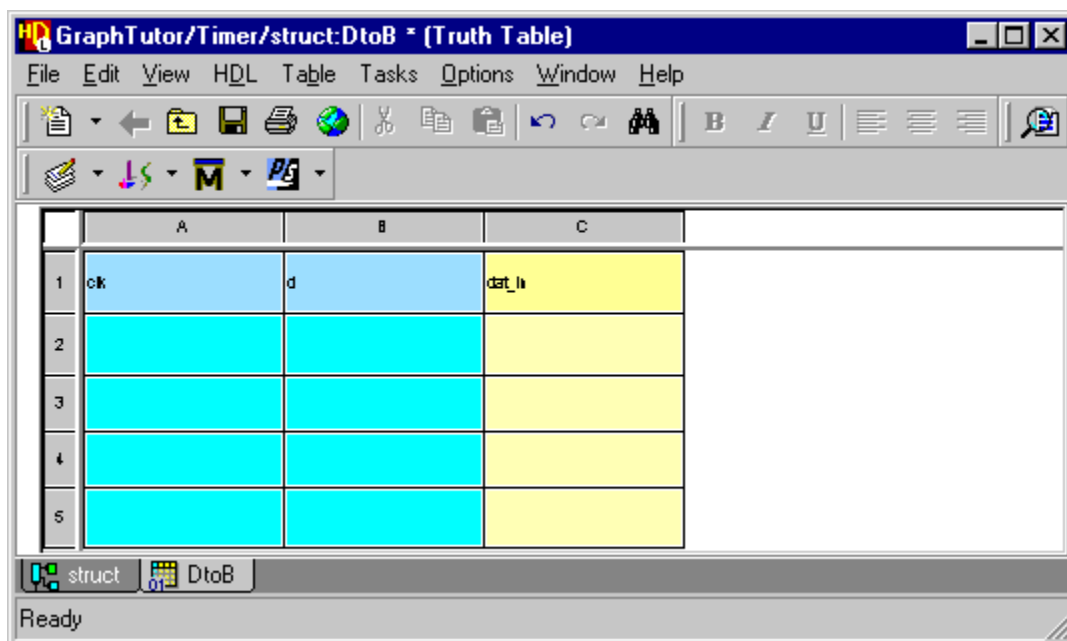
You have now defined a state diagram view for the *Control* block and a block diagram view for the *Counter* block.

The *Timer* design is completed by using a truth table to describe the *DtoB* embedded block.

1. Use the  button (or choose the **Open Up** option from the **File** menu) in the *Counter* block diagram to display its parent diagram (the *Timer* block diagram).
2. Double-click over the *DtoB* embedded block to display the Create Embedded View dialog box.
3. Select **Truth Table** and confirm the dialog box.

A new truth table (*GraphTutor/Timer/struct:DtoB*) embedded view is created and displayed in a tab window.

The table is initialized as a matrix of four rows with two input columns (for the *clk* and *d* inputs) and one output column (for the *dat_in* output).



Edit the Truth Table

4. Click the mouse in either of the blue input columns (A or B) and use the **Add Column** option from the popup menu to add another eight input columns (C to J).
5. Click the mouse in one of the blue input rows (rows 2, 3, 4 or 5) and use the **Add Row** option to add another seven rows (6 to 12) to the truth table.











You can add multiple rows or columns by selecting more than one cell to add the corresponding number of rows or columns.

6. Rename the input columns to represent each bit ($d[9]$ down to $d[0]$) of the d input bus and enter the value 1 in one row for each column. (The clk signal is not used and can be renamed.)
7. Rename the output column d_out and enter the following values:

```
4'b0000
4'b0001
4'b0010
4'b0011
4'b0100
4'b0101
4'b0110
4'b0111
4'b1000
4'b1001
4'b0000
```

To add text to a cell, click in the cell, enter the text and click in any other cell to confirm the entry. You can edit the text in a cell by double-clicking the cell and overwriting its contents or click again to edit individual text characters.

You can also use  button to copy and  button to paste text between cells and re-size the rows or columns by dragging the control sash in the non-scrolling area. You can format text in single or multiple selected cells by using the  (bold),  (italic) or  (underline) buttons and align text using the  (align center),  (align left) or  (align right) buttons.

The completed truth table should look similar to the picture below.

	A	B	C	D	E	F	G	H	I	J	K
1	d	d	d	d	d	d	d	d	d	d	d_out
2										1	4'b0000
3									1		4'b0001
4								1			4'b0010
5							1				4'b0011
6						1					4'b0100
7					1						4'b0101
8				1							4'b0110
9			1								4'b0111
10		1									4'b1000
11	1										4'b1001
12											4'b0000



The last (empty) row represents an ELSE condition and assigns the output expression a known value when none of the input expressions are true.

Set Truth Table Properties

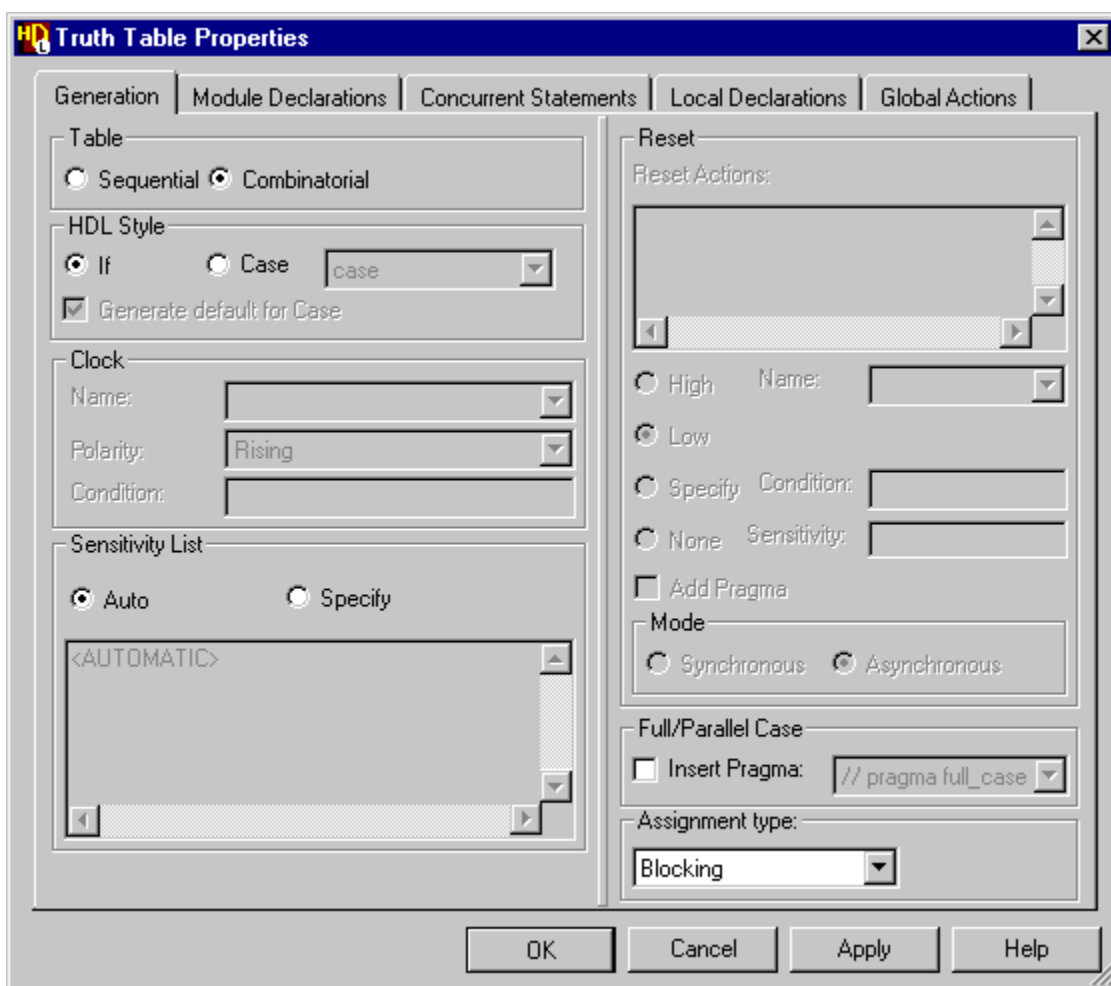
- Choose **Truth Table Properties** from the **Table** menu to display the Truth Table Properties dialog box. Select the **Concurrent Statements** tab and enter the following assignment:

```
assign dat_in = d_out;
```

This statement assigns the value of the truth table output (*d_out*) to the *dat_in* bus which is connected to the *Counter* block. This is necessary because Verilog compilation requires the truth table output to have type *reg* but the *dat_in* bus has the type *wire*.




The *d_out* signal must also be declared in the module for the design unit. Since the truth table is an embedded view of the *Timer* design unit, this must be done from the parent diagram and is described in “[Add a Module Declaration](#)” on page 2-61.

9. Select the **Generation** tab to display the HDL generation characteristics for the truth table. Notice that the Sensitivity List is set to **Auto**. When this option is set, the sensitivity list is automatically created by HDL generation. Check that the default options for **Combinatorial** and **If** style are set and use the button to confirm the truth table properties

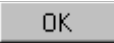
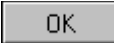


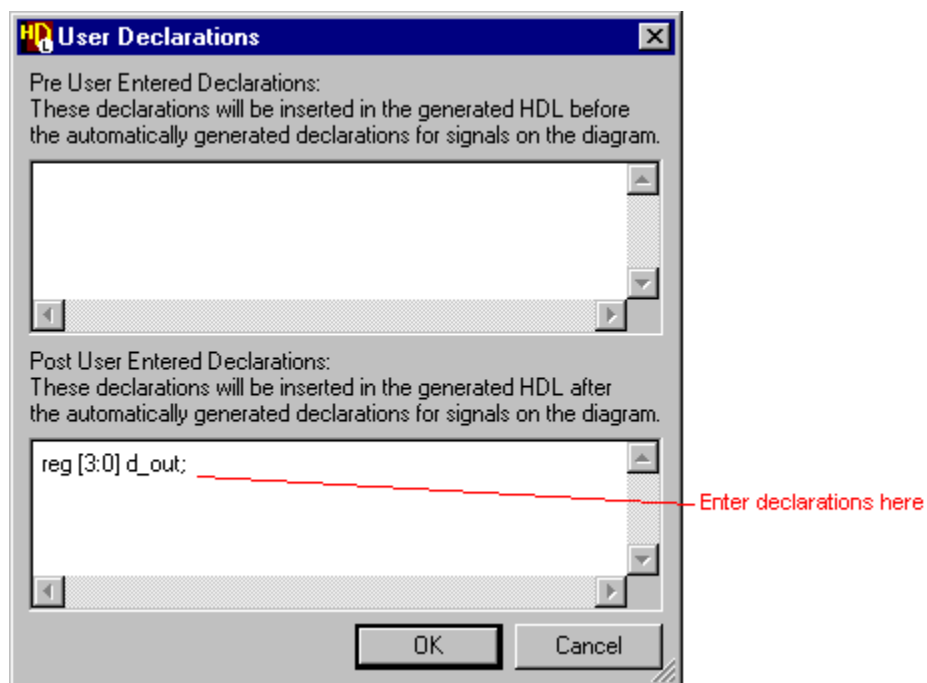
The tabs for Module Declarations, Local Declarations and Global Actions can be left empty for this tutorial.


Add a Module Declaration

10. Click the  *GraphTutor/Timer/struct* tab to display the *Timer* block diagram.
11. Double-click over the Declarations label on the block diagram (or use the  button) to display the **Declarations** tab of the block diagram Object Properties dialog box. Click the  button to display a free format entry box. Enter the following signal declaration in the Post User Entered Declarations section of the dialog box:

```
reg [3:0] d_out;
```

Click on the  button to confirm the user declaration and click the  button in the Block Diagram Object Properties dialog box to add the new user declaration on the diagram



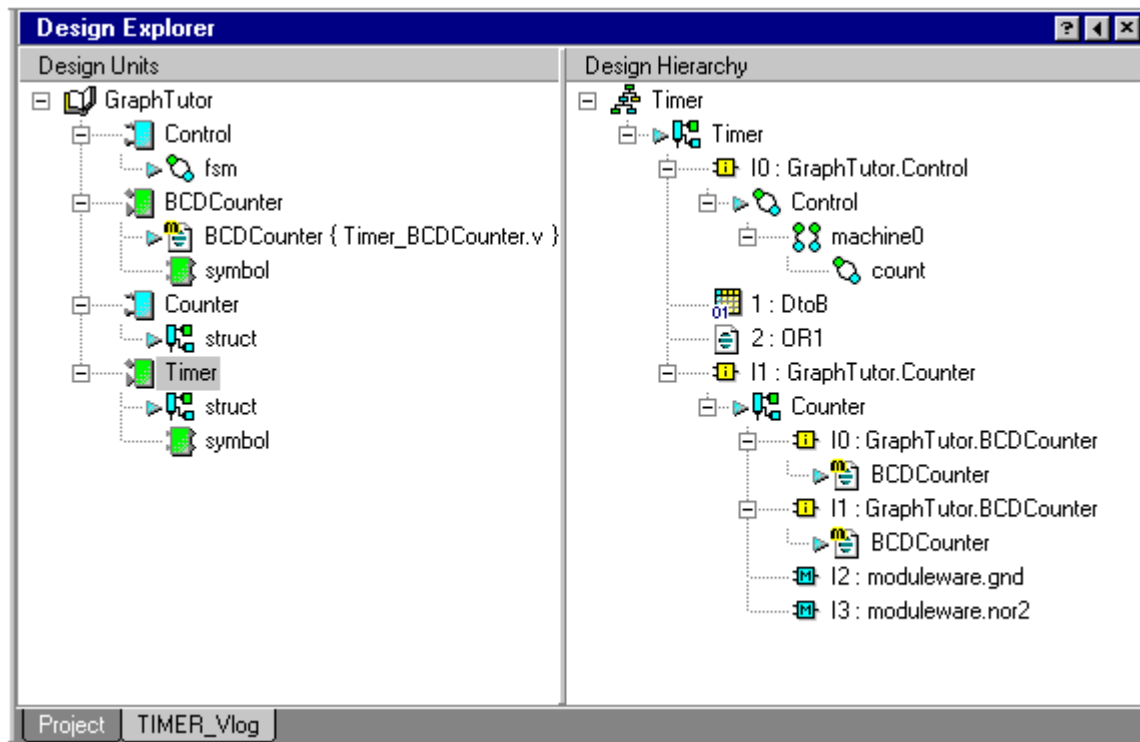
 The post user entered declarations are added to the declarations list on the diagram below the port and signal declarations.

12. Save the block diagram. This also saves the truth table since it is an embedded view within the same design unit as the *Timer* block diagram.

Browse the Timer Design

1. Examine the completed design in the design explorer *Design Units* and *Design Hierarchy* panes using the \oplus icons to expand each design unit.


The fully expanded design should look similar to the following picture:



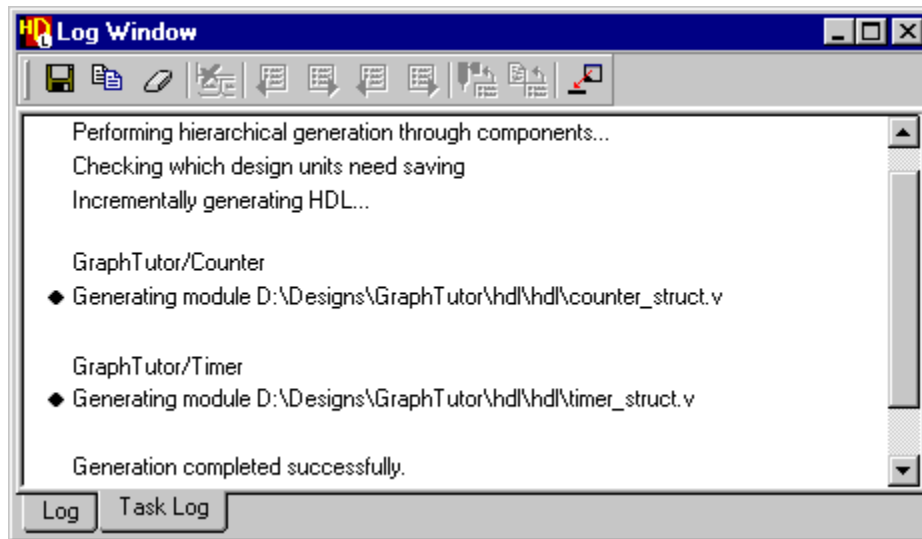
The expanded *Control* block design unit show that it contains a state machine (*fsm*). The *BCDCounter* component contains a Verilog module defined in the *Timer_BCDCounter.v* file. The *Counter* block contains a block diagram (*struct*). The *Timer* component contains a block diagram (*struct*) and a symbol describing its external interface.

Notice that you can expand the *Timer* component in the *Design Hierarchy* pane to display the complete hierarchy including the views describing the *BCDCounter* and the *Control* state machine. You can expand the state machine instance to show the child hierarchical state machine (*count*). You can expand the *Counter* instance to show it contains two instantiations of the *BCDCounter* plus the *gnd* and *nor* ModuleWare instances.





Generate HDL for the Hierarchy

2. Select the *Timer* design unit in the *Design Units* or *Design Hierarchy* pane of the design explorer window and choose the  button (or choose Run **Through Components** from the **Generate** cascade of the **Tasks** menu).

The progress of the HDL generation is monitored in a log window which includes any errors and warnings issued during generation.



The *Control* design unit generated earlier in this tutorial is not regenerated unless it has been changed. HDL is generated for all other design units. You are prompted to save any design units which have not been saved.

If there are any errors, you can move to the next error message using the  button or the previous error using the  button. You can display the source graphics corresponding to the error by double-clicking on the error message (or using the  button when the error is selected). Alternatively, you can use the  button when the error is selected to view the generated HDL. If your text editor supports a line number argument, the HDL line corresponding to the error is selected automatically.



The embedded truth table view is generated as part of the *Timer* design unit. If any syntax errors are issued for the truth table, the corresponding cells are highlighted. For example, if HDL generation encounters an expression which is not defined in the diagram interface, the corresponding input cells are highlighted.

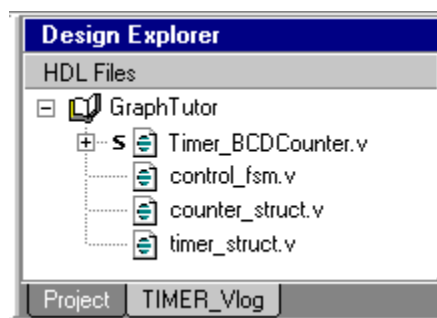
If there are no errors, you can use the toolbar buttons to view the Graphics or HDL corresponding to the “Generating...” message in the HDL Log window.



You can cross-reference to the graphics or HDL from any message which is marked by the ♦ icon in the log window.


Any files in the generated HDL directory can also be opened from the *HDL Files* view in the design explorer.

3. Use the  button (or choose HDL Files from the **Mode** cascade in the **View** menu) to display the *HDL Files* view in the design explorer.
4. Click on the  icon for the *GraphTutor* library and notice that the view is expanded to reveal the source file for the *BCDCounter* and the generated files for the *Control*, *Counter* and *Timer* design units:.



You can open any of these files by double-clicking on the generated file name or by choosing **Open File** from the popup menu.




You can also view the generated HDL files for the active diagram by using the  button in any editor window or when the diagram view is selected in the design explorer.

Edit the Timer Symbol

1. Use the  button (or choose the **Interface** option from the **Open** cascade of the **File** menu) in the *Timer* block diagram.

A symbol editor window is opened which shows the external interface for the *Timer* design as a default symbol. This symbol is used when the *Timer* design is instantiated as a component in a higher level design.

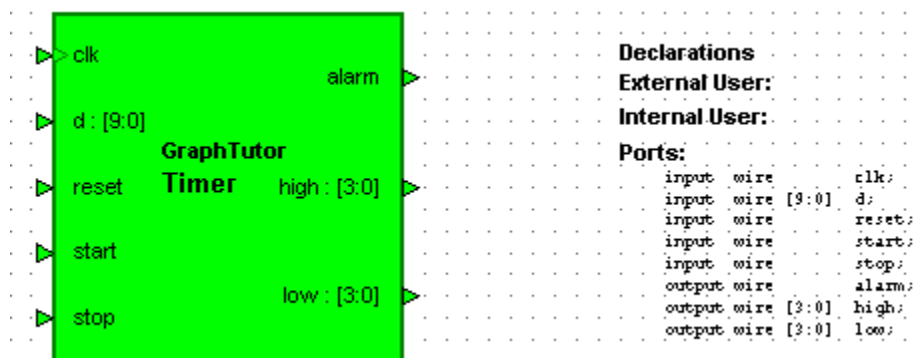
Ports representing the interface connections are shown in default locations but you can select and drag the ports to any location around the body of the symbol. You may also want to resize the symbol body.

-  You can redistribute the ports evenly by selecting the symbol body and choosing **Equidistant Ports** from the popup menu.

The full port declarations are shown as a list in the symbol editor and are not shown on the individual ports. Refer to “Changing the Visibility of Symbol Port Properties” in the *HDL Designer Series User Manual* for information about setting preferences to control whether type and bounds information is displayed or the properties visible for an individual port.

2. Select the *clk* port and choose **On** from the **Clock** cascade to apply an edge triggered clock indicator to the port.

The edited symbol should look similar to the picture below:



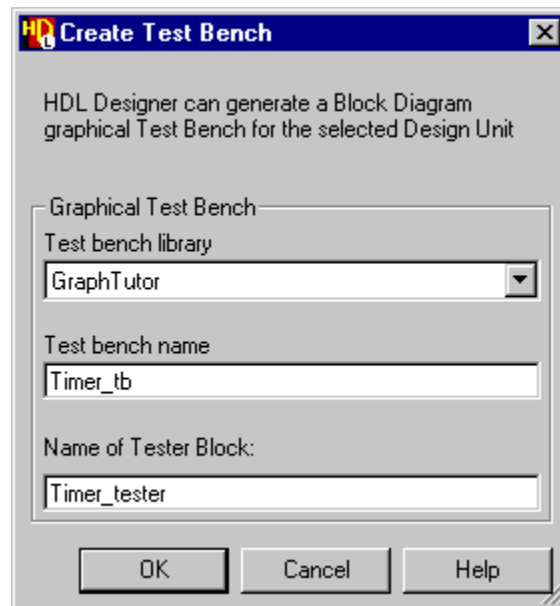
3. Close and save the symbol.

Create a Test Bench


1. Select the *Timer* design unit in the *Design Units* view of the design explorer and choose **Create Test Bench** from the **New** cascade in the **File** menu to display the Create Test Bench dialog box.

Notice that the dialog box defaults to the *GraphTutor* library, with the test bench name derived by adding *_tb* to the *Timer* design unit and the default tester block by adding *_tester*.

2. Confirm the dialog box by clicking the button.



Notice that a *Timer_tb* design unit is added to the *Design Units* displayed in the design explorer window.

3. Click on the  icon for the *Timer_tb* design unit in the design explorer to expand the view and reveal that a symbol and block diagram view (with the default name *struct*) have been created.
4. Double click on the *struct* view to open the block diagram.

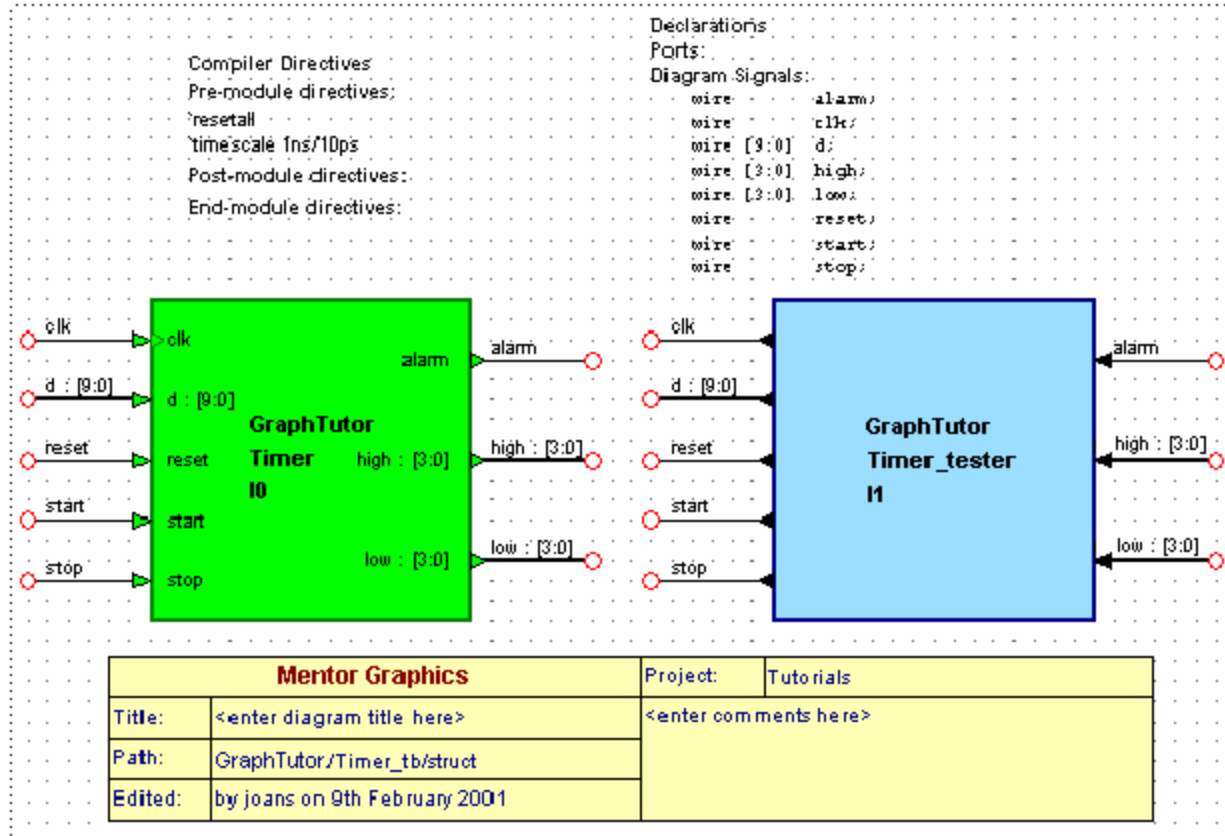
Notice that the *Timer* design has been instantiated as a component with stub signals connected to the input and output ports. The adjacent *Timer_tester* block has corresponding output and input ports which are implicitly connected by name to the stub signals on the component.



The port locations are identical on the *Timer* component and *Timer_tester* block but their directions are reversed. Hence the output ports are on the left and input ports on the right of the block.

All the signals and buses are declared as diagram signals since a test bench typically has no external ports.

The test bench block diagram should look similar to the following picture.




It is not necessary to explicitly connect signals and buses between each port on the component and the tester block as these are implicitly connected by name.

Import the Tester Design Unit

The *Timer_tester* block should provide test stimulus to the *Timer* design unit and monitor its output. This can be achieved by using a flow chart or HDL text view.

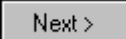

For this tutorial, the *Timer_tester* design unit can be imported from the *examples* installation subdirectory using a similar procedure to that used to “[Import the BCDCounter Design Unit](#)” on page 2-44.

The *Timer_tester* design unit can optionally be created as a flow chart. Refer to the alternative procedures for [Creating a Verilog Flow Chart](#) which are provided in [Chapter 4](#).

1. Use the  button in the design manager window (or choose **HDL Import** from the **HDL** menu) to display the HDL Import wizard.
2. Choose **Specify HDL files** in the first page and select the *Timer_tester.v* file in the Specify HDL Source Files page of the wizard.



The previous HDL import directory and files are saved as preferences. You should only need to remove the *Timer_BCDCounter.v* file and add the *Timer_tester.v* file.


3. Use the  button to page through the HDL Import wizard and use the  button on the last page to import the *Timer_tester* design unit as a HDL text view.

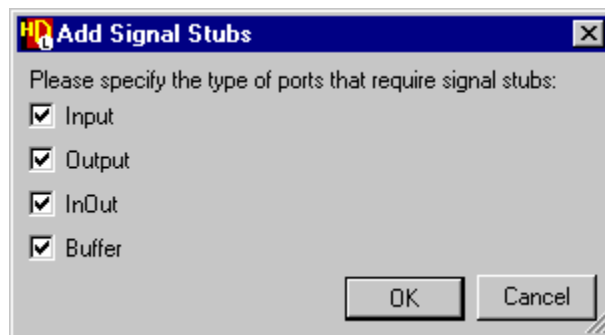
The following completion message should be displayed in the HDL log window:

```
** Importing 'D:\Designs\GraphTutor\hdl\Timer_tester.v'
HDL Import complete
```

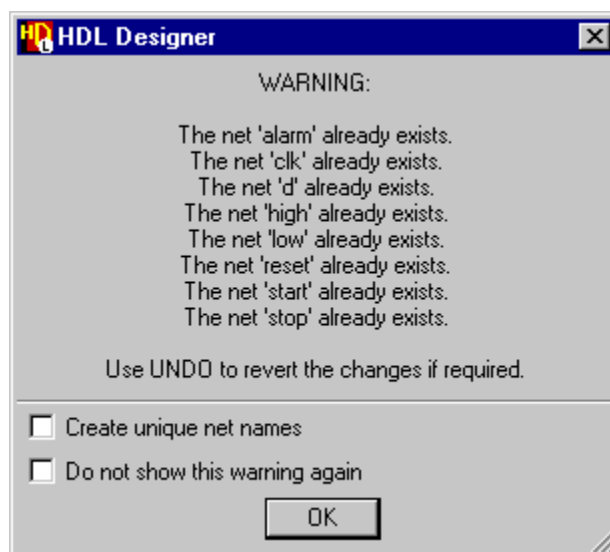
```
-----
1 file imported to 1 library
-----
```

Instantiate the Imported Tester

4. Re-open the *Timer_tb* block diagram if it is not still open. Select the *Timer_tester* block including all the connected signal stubs and delete them using the **Del** key.
5. Use the  key to display the Component Browser. Select the *GraphTutor* library and instantiate the imported *Timer_tester* component in the block diagram.
6. Select the *Timer_tester* component and choose **Add Signal Stubs** from the popup menu to display the Add Signal Stubs dialog box:



7. Click the **OK** button to confirm that you want to add signal stubs to all port types. You are warned that the nets *alarm*, *clk*, *d*, *high*, *low*, *reset*, *start_in* and *stop* already exist on the diagram.



8. Click the button to acknowledge the warning. The signal stubs are added when you accept the warning and the matching signals are implicitly connected by name.
9. Complete the test bench by editing the title and comments in the title block and using the button to add a panel around the components. This panel can be used to set the diagram view when you simulate your test bench later in this tutorial.

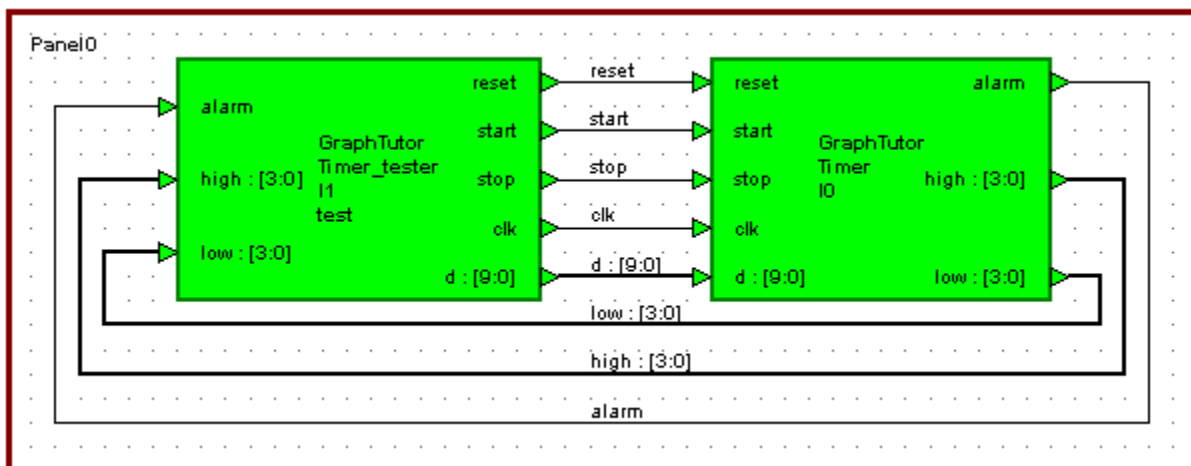
The matching nets on the two components are now implicitly connected by name. However, you may like to connect them explicitly by choosing **Autoconnect** from the **Autoroute** cascade of the **Diagram** menu.



Using this command with nothing selected should automatically route all nets on the diagram. If any nets are selected, then only the matching nets in the selection are routed.

It may also be necessary to move some of the component ports in order to avoid cross-overs.

The following example shows a test bench which has been explicitly connected:

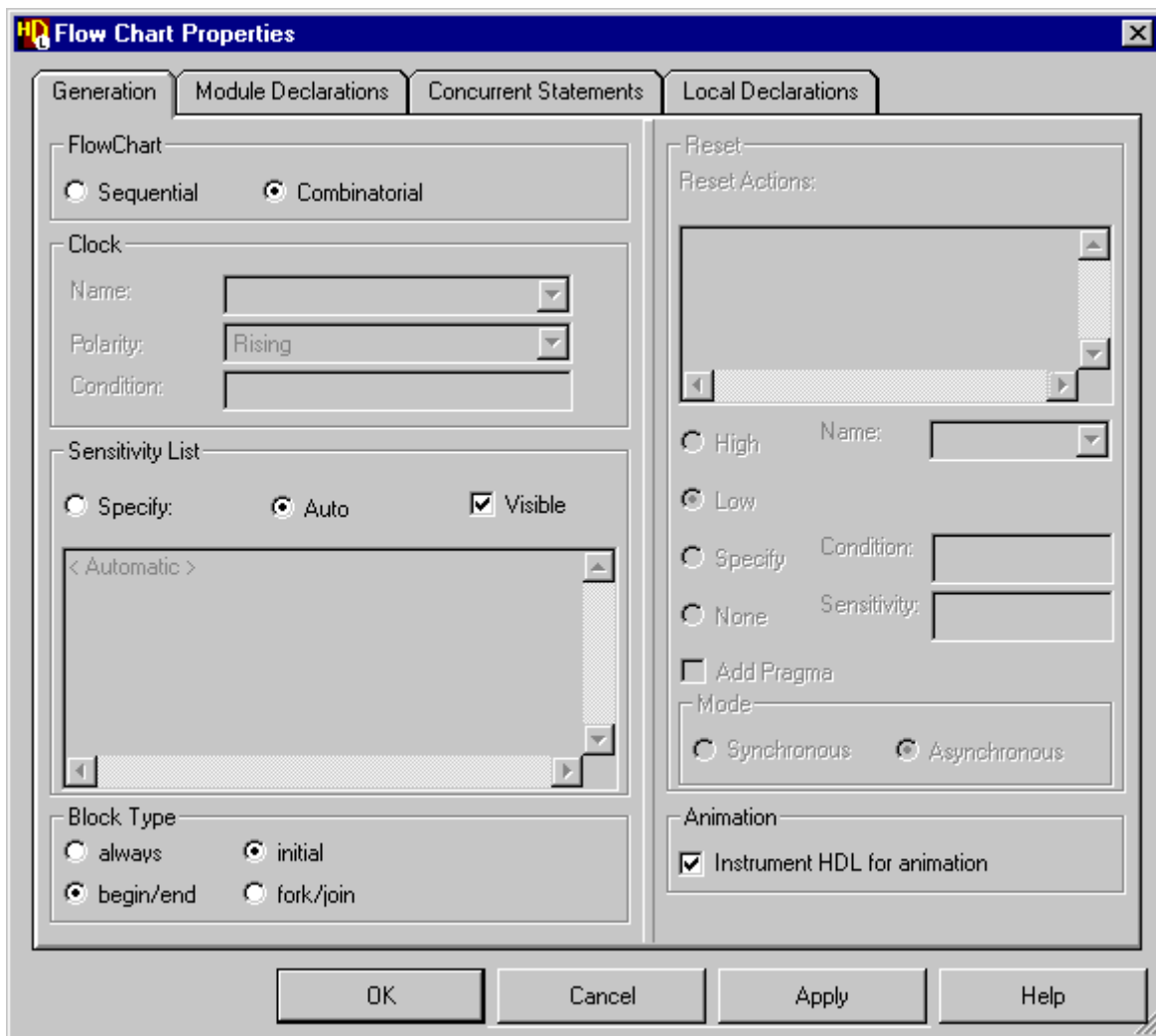


Generate HDL for the Test Bench

- If the *Timer_tester* component is described by a flow chart, double-click on its instance in the test bench to open the view and choose **Flow Chart Properties** from the **Diagram** menu to display the Flow Chart Properties dialog box.

i If you have recovered the *Timer_tester* component as a HDL text view, all properties for the view are defined in the HDL text and do not need to be set using a dialog box.



- Choose the **Generation** tab and check that the **Combinatorial** option is set.




12. Set the Block Type options to **initial** and **begin/end**. This flow chart represents a test bench and the HDL code must be generated using initial style code.
13. If a ModelSim or NC-Sim simulator is available, set the **Instrument HDL for animation** option. This option adds additional code to the generated HDL which is used for flow chart animation later in this tutorial.



The additional animation code is surrounded by translation control pragmas which ensure that it is ignored by downstream synthesis tools.


14. Set the **Auto** option for the Sensitivity List and use the  button to confirm the dialog box.
15. Select the *Timer_tb* design unit in the design explorer and choose the  button (or choose **Run Through Components** from the **Generate** cascade of the **Tasks** menu).

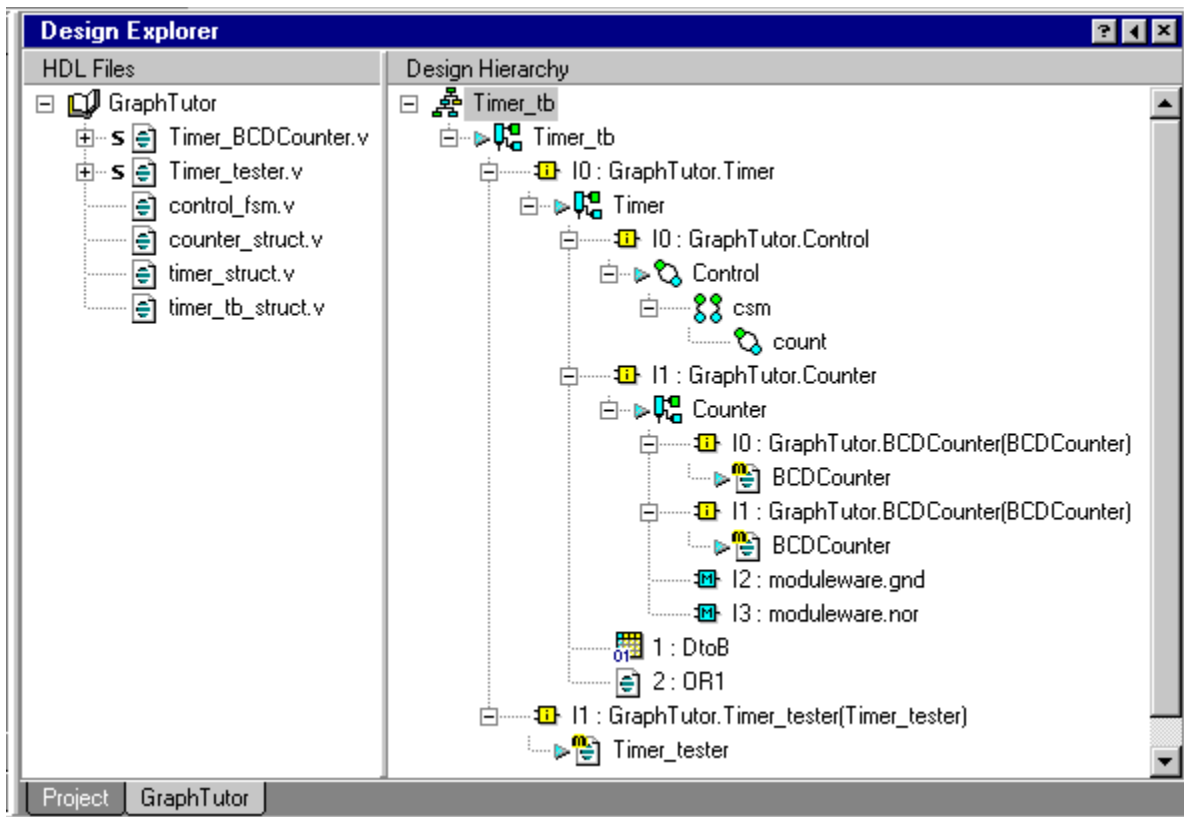
All views to be generated must have been saved and if you have not saved the test bench you are prompted whether to continue. Confirm the dialog box by clicking the  button.

Verilog is generated for the test bench and tester (if it is described by a flow chart) views but not the *Timer* design hierarchy since these views were generated earlier in this tutorial and do not need to be regenerated unless they have been modified.

The progress of the HDL generation is monitored in a log window which includes any errors and warnings issued during generation. If there are any errors, you can display the corresponding graphics by double-clicking on the error message as described in the section [“Generate HDL for the Hierarchy”](#) on page 2-63.

Browse the Completed Design

1. Select the *Timer_tb* design unit and choose **Show Hierarchy** from the popup menu to display the *Design Hierarchy* pane of the design explorer.
2. Select the *Timer_tb* design unit in the *Design Hierarchy* pane and choose **Expand All** from the popup menu or use the \oplus icons to expand the full hierarchy which also includes the hierarchy of the *Timer* design.
3. Use the  button to view the *HDL Files* pane which should now show the *Timer_BCDCounter.v* and *Timer_tester.v* source files plus the generated files for *control_fsm.v*, *counter_struct.v*, *timer_struct.v* and *timer_tb_struct.v*.



You can use the *Logical Objects* pane to explore the design in detail. Refer to the the [HDL Designer Series User Manual](#) for more information.

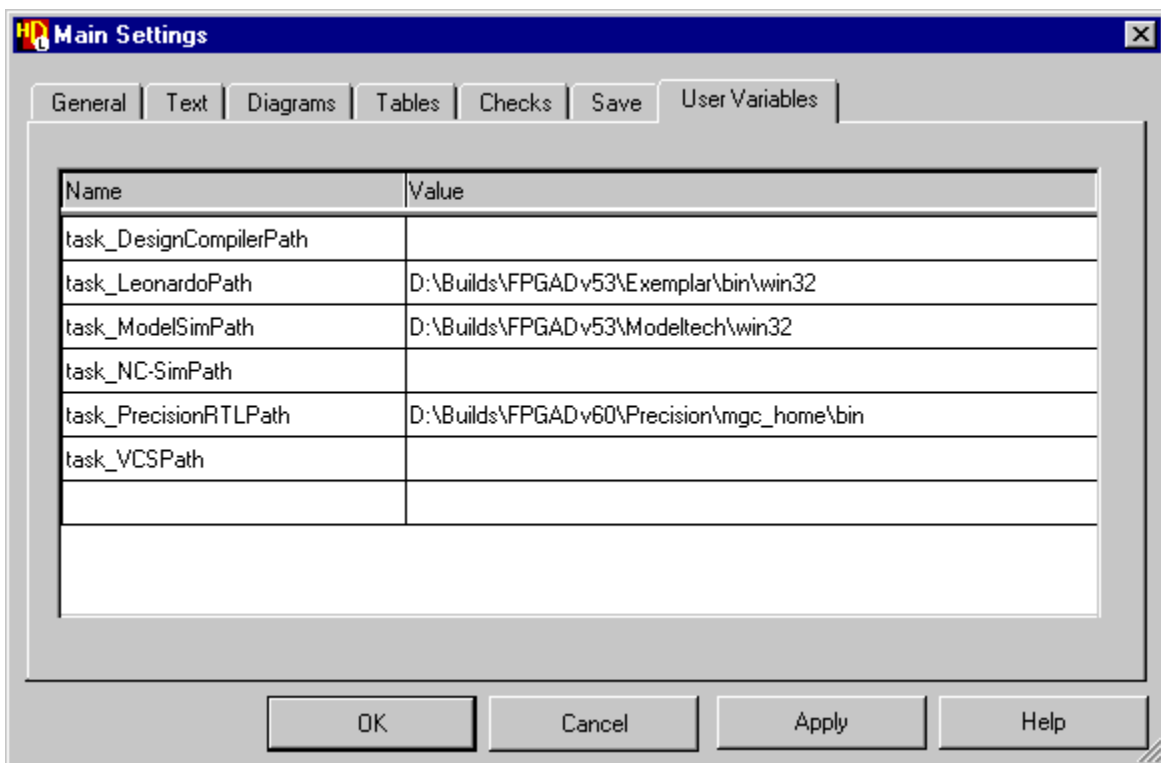
Setup the Downstream Tools

For this tutorial, it is assumed that ModelSim simulator and LeonardoSpectrum synthesis tools are available. You can alternatively prepare data for use with other downstream tools.



If you have installed the HDL Designer Series tool as part of FPGA Advantage, the ModelSim simulator and LeonardoSpectrum synthesis tools will already have been set up automatically.


1. Choose **Main** from the **Options** menu to display the Main Settings dialog box and choose the **User Variables** tab:



User variables are provided for the Design Compiler, LeonardoSpectrum, ModelSim, NC-Sim, Precision Synthesis and VCS (or VCSi) tools.

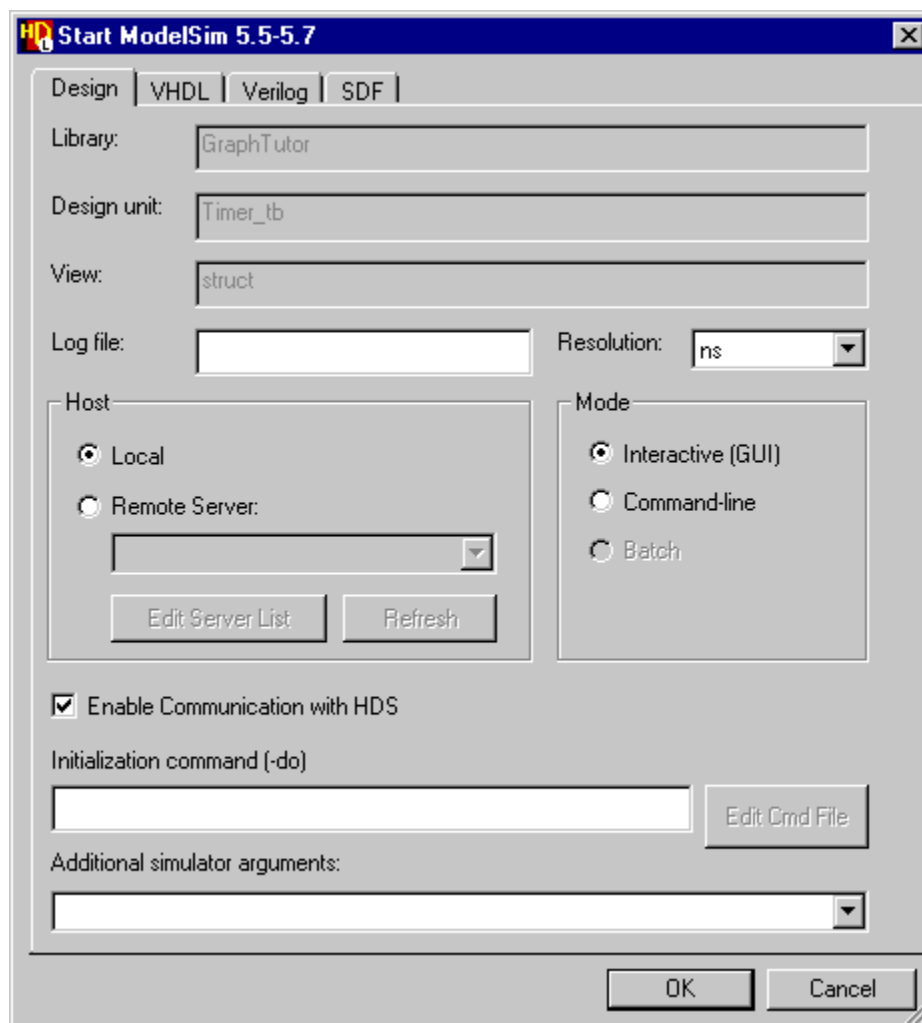
2. Enter the pathname to the executables for any of the downstream tools which are available on your system and confirm the dialog box.

Run the ModelSim Flow

1. Select the *Timer_tb* design unit in the design explorer and use the  button to run the ModelSim flow through components.

HDL is regenerated for any modified graphical views and the entire design is compiled. The progress of the compilation is shown in the HDL Log Window. If there are any errors, you can display the corresponding source diagram by double-clicking on the error message as described in the section [“Generate HDL for the Hierarchy”](#) on page 2-63.

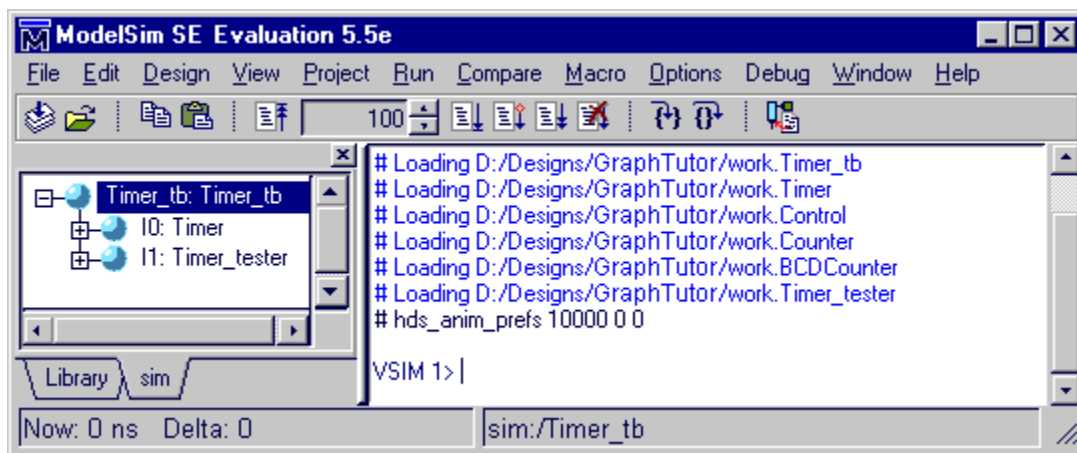
If there are no errors, the Start ModelSim dialog box is displayed:



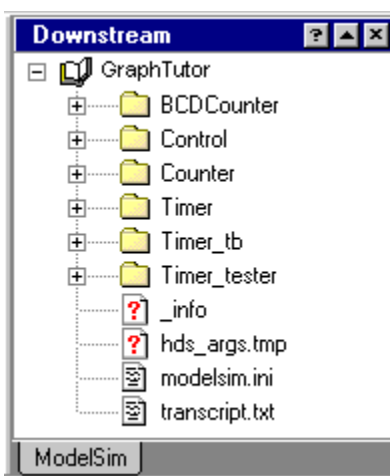
2. Accept the default resolution and check that the **Interactive (GUI)** and **Enable Communication with HDS** options are selected. The other entry fields can be left in their default state.
3. Use the button to confirm the Invoke Settings dialog box.

The simulator is invoked and issues a number of loading messages ending with:

```
# hds_anim_prefs 10000 0 0.
```



Note that library mapping is automatically created for your downstream data and the compiled objects can be displayed in the downstream browser by selecting **Downstream** from the **SubWindows** cascade of the design explorer **View** menu and choosing the ModelSim tab:



Setup the Simulator Windows



An additional Simulation toolbar is available in the block diagram, flow chart and state diagram when a simulator is invoked to support cross-probing between the simulator and source design objects in the HDL Designer Series tool. This toolbar is normally displayed automatically at the bottom of the diagram window but can be undocked, moved, docked or hidden in the same way as the other toolbars.

For example, the following picture shows a toolbar that has been undocked from the editor window and re-arranged as two rows of tool buttons:



4. Display the *Timer_tb* block diagram and choose **View Panel** from the **View** menu to display the panel you added earlier in this tutorial.




i If you added more than one panel, a dialog box is displayed for you to choose the panel to display.

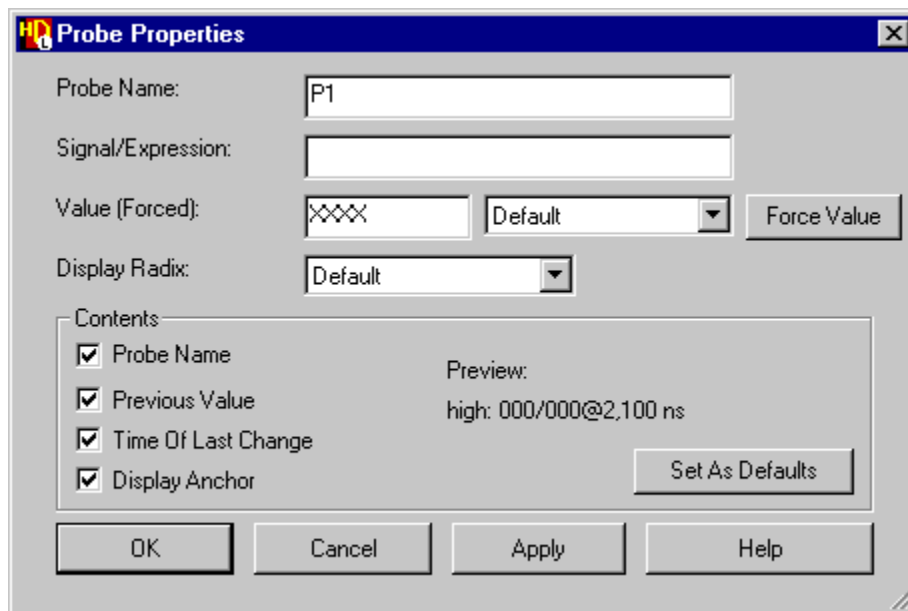
5. Select the signals *alarm*, *clk*, *d*, *reset*, *start*, *stop*, *high* and *low*. Use the  button from the Simulation toolbar or choose **Add Wave** from the **Display** cascade in the **Simulation** menu and notice that the simulator Wave window is automatically opened with these signals loaded.
6. Use the  button from the Simulation toolbar or choose **Add List** from the **Display** cascade in the **Simulation** menu to add the selected signals to the simulator list window.


i You can optionally add signals to the simulation log without displaying them in the Wave or List window by choosing **Add Log** from the **Display** cascade of the **Simulation** menu.

You can also open any other simulator window by choosing from the **View** cascade of the **Simulation** menu. For example you may wish to use this menu to open the ModelSim **Source** and **Structure** windows.

Add Simulation Probes

- Set simulation probes on the *low* and *high* signals by using the  button or by choosing **Add** from the **Probes** cascade of the **Simulation** menu. The probes are shown by  icons with text displaying the current values.
- Select the probe on the *high* signal and use the  button in the Simulation toolbar to display the Probe Properties dialog box.



- Change the probe name from the default net name to *P1* and set the Contents options for **Probe Name**, **Previous Value** and **Time of Last Change**. Use the  to update the probe display on the block diagram.
- Repeat this procedure for the probe on the *low* signal renaming this probe as *P2*. Both probes should now display the probe names, current and previous values (initially unknown) and the current simulation time (initially 0 nanoseconds):



The probes are connected to the signal nets by an anchor and can optionally be moved to improved diagram clarity.

Enable Animation

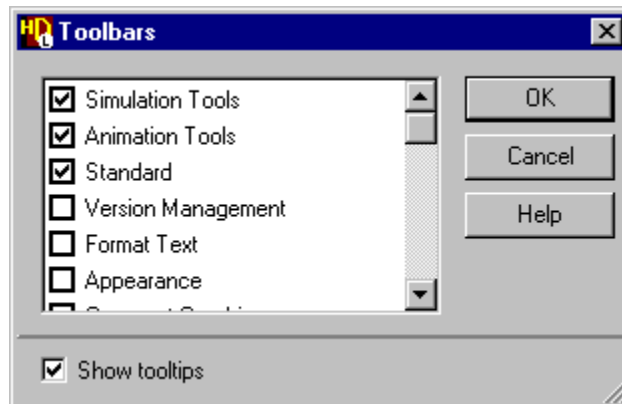
- Open the *Control* state diagram and its child hierarchical state diagram. Use the **View Panel** command in each tab to adjust the window view to the panel areas you defined earlier in this tutorial.

i If you have created a flow chart view for the *Timer_tester* view, it can be animated in a similar way to that described below for the state diagrams.

Notice that an additional Animation toolbar is available in the state diagram (and flow chart) windows when the simulator is invoked. This toolbar is normally docked at the bottom of the diagram window next to the Simulation toolbar but can be moved independently.




The editing toolbars are not usually required during simulation and animation. You can hide or show toolbars using the Toolbars dialog box which is displayed by choosing **Settings** from the **Toolbars** cascade of the **View** menu.






- For example, use the dialog box to hide the Format Text, Appearance, Comment Graphics and Arrange Object toolbars.

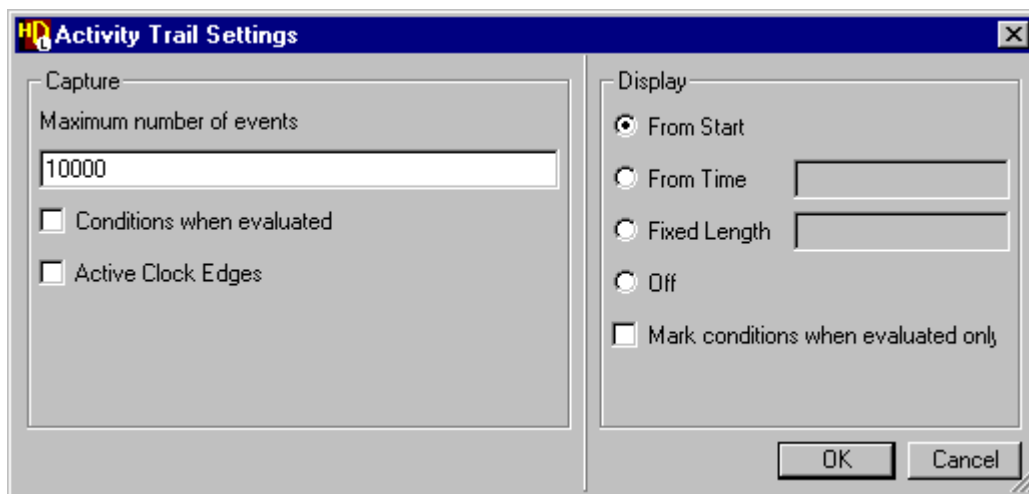
i Individual toolbars can also be hidden by unsetting the corresponding options in the **Toolbars** cascade of the **View** menu.


13. Use the  buttons from the Animation toolbars (or choose **Data Capture** from the **Animation** menus) in each state diagram or flow chart you want to animate. (It is not necessary to repeat this command for each hierarchical or concurrent view since these are considered part of the same diagram.)


Notice that all objects (except the start state, exit and entry points in the state diagram are redrawn with white fill. The start state is drawn with red fill to indicate that it is the current animation state.

-  This animation view is automatically displayed when you set data capture but can be toggled at any time by using the  button (or by toggling the **Show Animation** option in the **Animation** menu). You can also set data capture for all instances in the current simulation hierarchy by choosing **Global Capture On** from the **Animation** menu.

14. Use the  button from the Animation toolbar (or choose **Activity Trails** from the **Animation** menu) in the flow chart or state diagram window and choose the **From Start** option in the Activity Trail Settings dialog box.



15. Use the  button to confirm the Activity Trail Settings dialog box.


-  The activity trail is applied to all flow chart or state diagram windows in the current simulation.



16. Use the  button or choose **Add State Variable to Wave Window** from the **Display** cascade of the **Simulation** menu in the state diagram.

This command monitors the state machine variable by a signal corresponding to the current state of the state machine.



17. Use the  button or the **Add State Variable to List Window** command to add the same signal in the List window.


Simulate the Design


18. Run the simulator for the default timestep (100 nano-seconds) by using the  button or by choosing **For Time** from the **Run** cascade of the **Simulation** menu in the block diagram, flow chart or state diagram.

-  The default timestep can be changed by using the  button to select from a list of alternative timesteps or choose a user specified timestep.


Notice that the *Initialization* action box is highlighted red in the animated flow chart for the *Timer_tester* and the signal values are updated in the simulation probes on the block diagram and in the simulator Wave and List windows.

19. Run the simulator for another default timestep and notice that the P1 probe is shown by a yellow icon because its value is unchanged since the last simulator timestep.
20. Select the *stop* signal in the *Timer_tb* block diagram and set a breakpoint by using the  button or choosing **Add** from the **Breakpoints** cascade of the **Simulation** menu. The breakpoint is shown by a  icon.

-  You can optionally add a simulation probe to the *stop* signal using the procedure described in [“Add Simulation Probes” on page 2-78](#).


21. Run the simulator until there are no more events scheduled by using the  button from the Simulation toolbar or by choosing **Forever** from the **Run** cascade of the **Simulation** menu.

The simulation should run until the breakpoint is encountered when the value of the *stop* signal value (shown in the Wave and List windows) changes to '1'. Notice that the *Store* action box is highlighted in red as the current step in the animated flow chart and the *counting* state is entered in the child animated state machine for the *count* state.



22. Use the  button or choose **Continue** from the **Run** cascade of the **Simulation** menu to continue the simulation until the *stop* signal changes back to zero.

Notice that the *suspended* state is entered in the animated state machine and the following note is issued in the main simulation window:

```
# Count suspended correctly
```


23. Run the simulator to the next change of the *stop* signal by using the  button and notice the message:


```
# Alarm asserted correctly
```

24. Use the  button in the test bench block diagram to delete the breakpoint on the *stop* signal and complete the simulation by using the  button. The simulation should run to completion and issue the message:


```
# Timer test completed
```

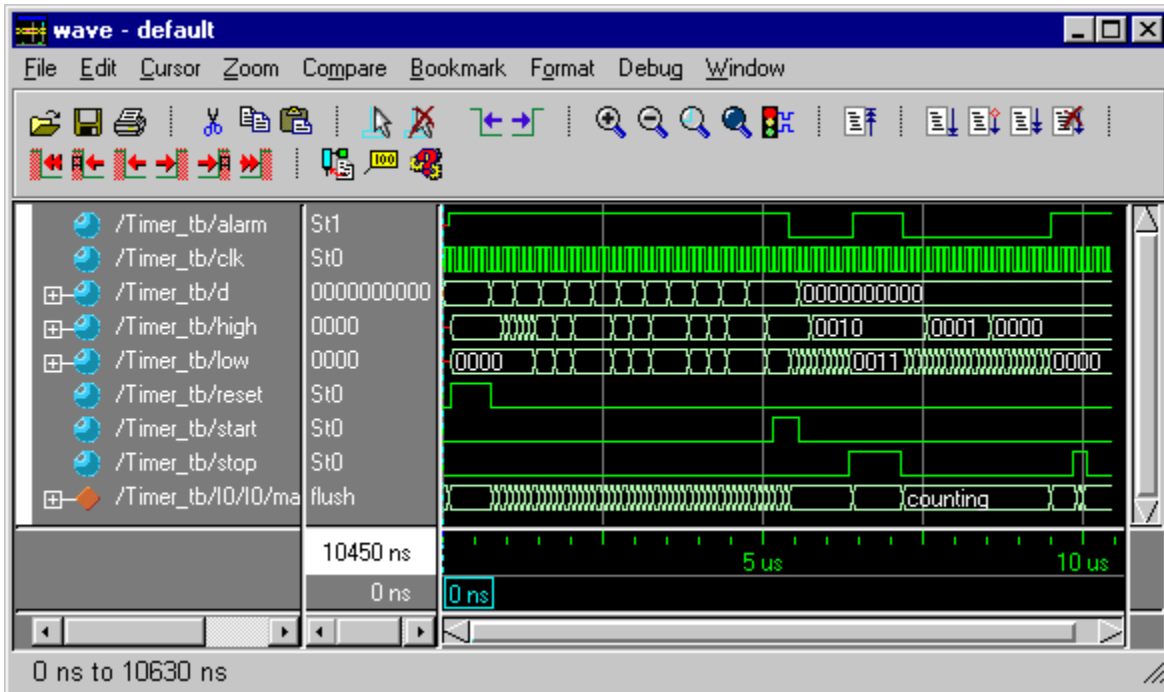
Examine the results displayed in the Wave window and ensure that the simulation has performed correctly by comparison with the example waveforms shown on the next page.


If any Verilog errors are discovered, correct your design and use the  button to regenerate and recompile the modified diagram. If the diagram is successfully generated and compiled, the flow button automatically restarts the simulation for the current simulation hierarchy.

Alternatively, you can use the  button or choose **Restart Simulator** from the Simulation menu to repeat the simulation.




You can use the  button or choose **Highlight Object** from the **Display** cascade of the simulation menu to highlight all occurrences in the simulation windows of a signal corresponding to any selected graphical object.













i You can display the full simulation waveform by using the  button or choosing **Zoom Full** from the Wave window **Zoom** menu.

Review the Animation

Notice how the animation activity trail in the animated diagrams is highlighted in blue and the exit break point for `Timer test completed` is indexed on the flow chart Verilog module displayed in the simulator Source window.

25. If you have enabled animation for the *Timer_tester* flow chart, use the  button or choose **Link Diagrams** from the **Animation** menu to link the animated diagrams. When the diagrams are linked, the animation review commands are applied to all the animated flow charts and state diagrams in the simulation hierarchy.

You can review the animation by using the , , ,  or  buttons (or by choosing **Goto Next**, **Goto Previous**, **Goto Time**, **Goto Start** or **Goto Latest** from the **Animation** menu).


The  and  buttons step through each object in the flow chart. However in the state diagram, you can set options in the **Animation** menu (or from a pulldown menu on the toolbar button) to move by change of state , simulation event  or change of clock edge . Notice how the toolbar button icon changes to indicate the current mode of movement.


You may want to change the activity trail to display the trail from a specified time or specify a fixed length. The current simulation step (or current state and last transition taken in an animated state diagram) is always shown in red and the previous step (or previous state and transition) in yellow. All other visited objects included in the activity trail are shown in blue. Remember that you can open down the hierarchical action box or hierarchical state to view the animation in the child diagrams.

26. Use the toolbar buttons to move forwards and backward through the animation.
27. Compare the animation with the results displayed in the Waveform window and ensure that the simulation has performed correctly by comparison with the example waveforms on the previous page.

Debugging From ModelSim



If you are using ModelSim, additional toolbar and menu commands are provided for cross-probing to the graphical views. These include the following:


A  button is available in the Animation toolbar and a **Cause** option from the **Animation** menu. These commands can be used to move the ModelSim Wave and List windows to the current animation time.


A corresponding  button and **Cause** option (in the **Debug** menu) are available in the ModelSim Wave window. These commands can be used to update all currently open animation windows to the simulation step or event immediately preceding the time marked by the Wave window cursor.


A **Cause** command is also added to the **Debug** menu in the simulator List window which can be used to update all open animation windows to the simulation step or event corresponding to the selected line in the List window.

You can also move through the simulation by using a *ModelSim* cursor. The state machine animation and the values displayed in the simulation probes are setup to track the cursor by default.

 These options can be enabled or disabled by using the  button or by setting options in the state diagram **Display** menu and the block diagram **Probes** menu.

A  button is added to the toolbar (and a **Trace To HDS** option is added to the **Debug** menu) in the Source window which can be used to open the graphic window showing the source object corresponding to the line of HDL code under the cursor.


The **Trace To HDS** command is available in the **Debug** menu for the *ModelSim* Wave, List and Signals windows (or using the  button in the Wave window). It can be used in these windows to display the source object where the selected signal is declared. (Typically, this will be the test bench or top level block diagram for the design being simulated.) You can also trace a signal to HDS from the Wave or Signals windows by double-clicking on the signal.

A  button and **Trace To HDS** command are available from the **Debug** and popup menus in the main simulation window. A **Trace To HDS** command is also added to **Debug** menu in *ModelSim* Structure window. These commands can be used to open the source object corresponding to the selected instance.

Refer to “Cross Probing from *ModelSim*” and “Using the *ModelSim* Source Window” in the *HDL Designer Series Graphical Editors User Manual* for more information about the cross-probing commands added to *ModelSim* when it is used with a HDL Designer Series tool.

28. Exit the simulator (by choosing **Quit** from the **File** menu in the main *ModelSim* window). Any other simulator windows are automatically closed.
29. Close all graphic editor windows by choosing **All Editor Windows** from the **Close** cascade of the **File** menu in the design manager.

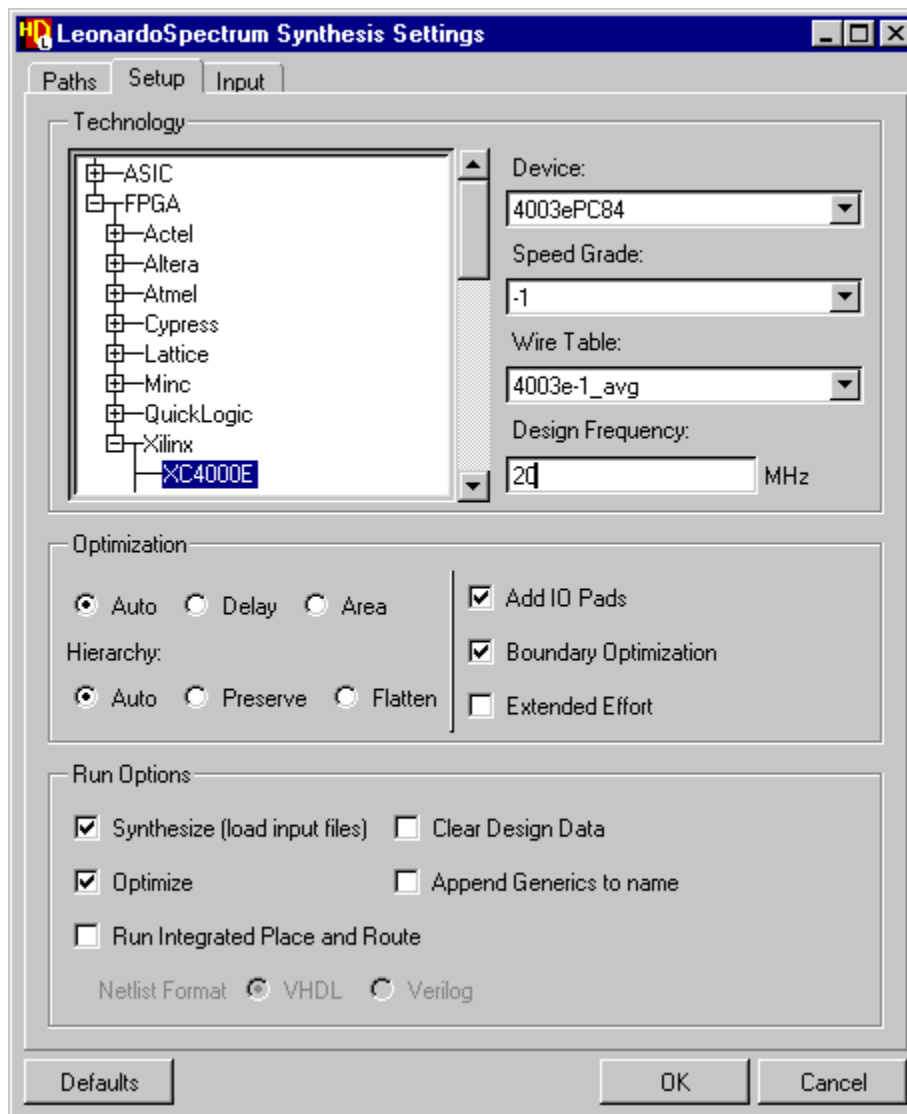
Run the LeonardoSpectrum Flow


1. Select the *Timer* design unit in the design explorer and use the  button to run the LeonardoSpectrum flow through components.



The tester design unit used in the test bench contains unsynthesizable HDL. Ensure that you have selected the *Timer* and not the *Timer_tb* design unit in the design explorer.

The design will be regenerated if necessary and the LeonardoSpectrum Synthesis Settings dialog box is displayed:



- Use the  icons to expand the FPGA technologies list and select the technology of your choice. For example, Xilinx XC4000E.



If you have an Exemplar level 3 license, ASIC and FPGA technology libraries are available. If you have a level 2 license, only the FPGA libraries are available.

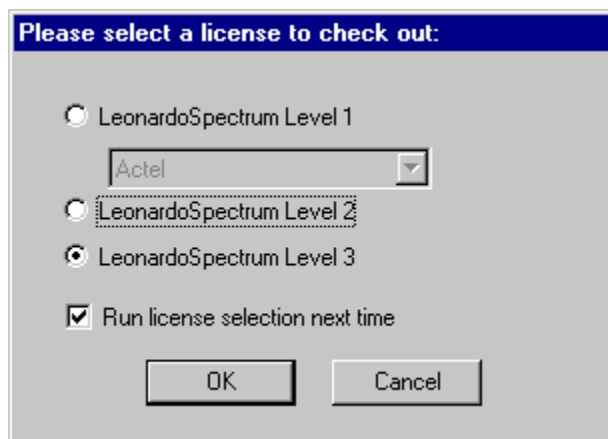
The Device, Speed Grade and Wire Table fields are automatically selected when you have chosen a technology.

- Complete the dialog box by entering a clock frequency (for example 20 MHz). All other options can be left with their default values.

The options in the other tabs can also be left with their default values.

- Use the  button to confirm and close the dialog box.

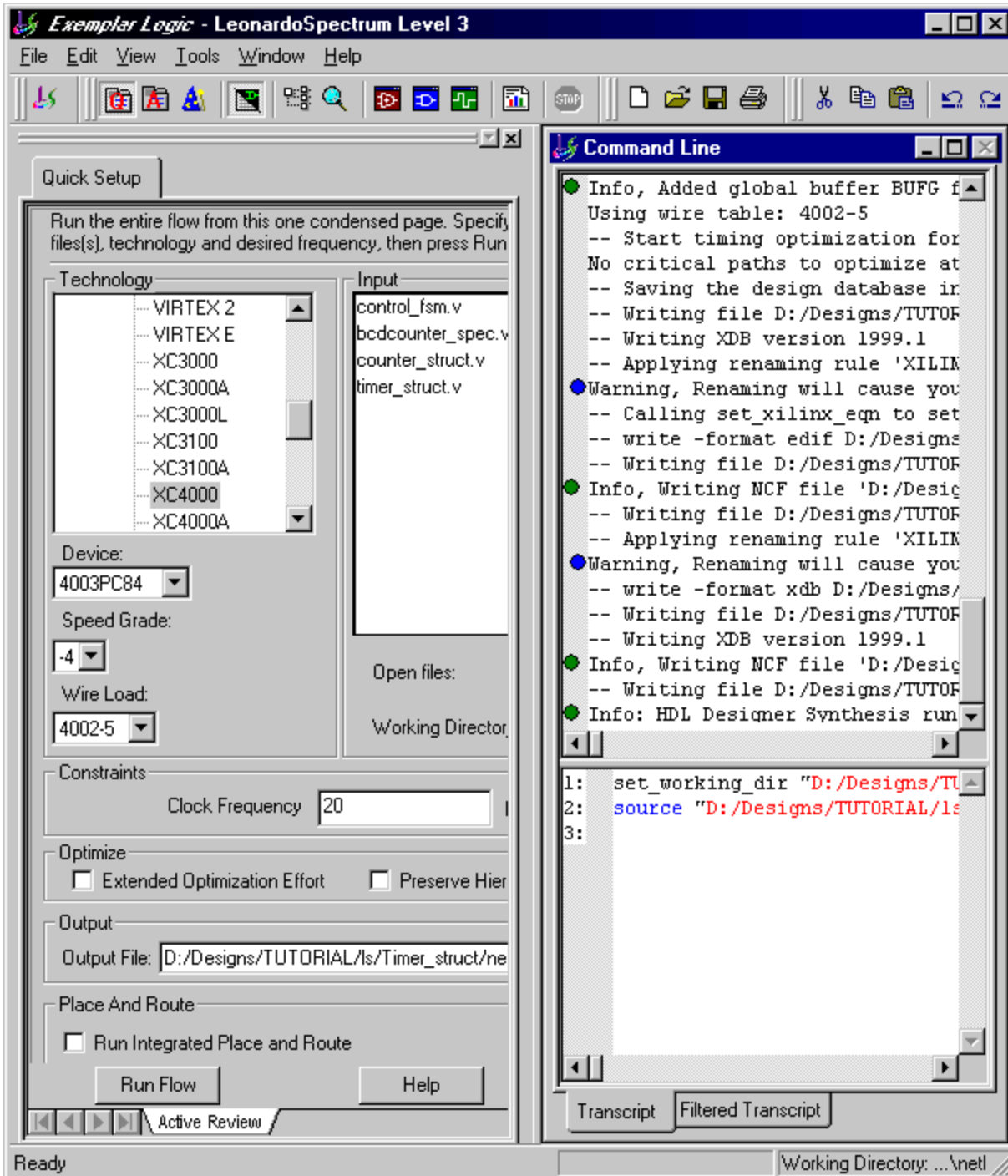
When you confirm the invoke settings, LeonardoSpectrum is invoked and you are prompted to select an Exemplar level 1, 2 or 3 license.





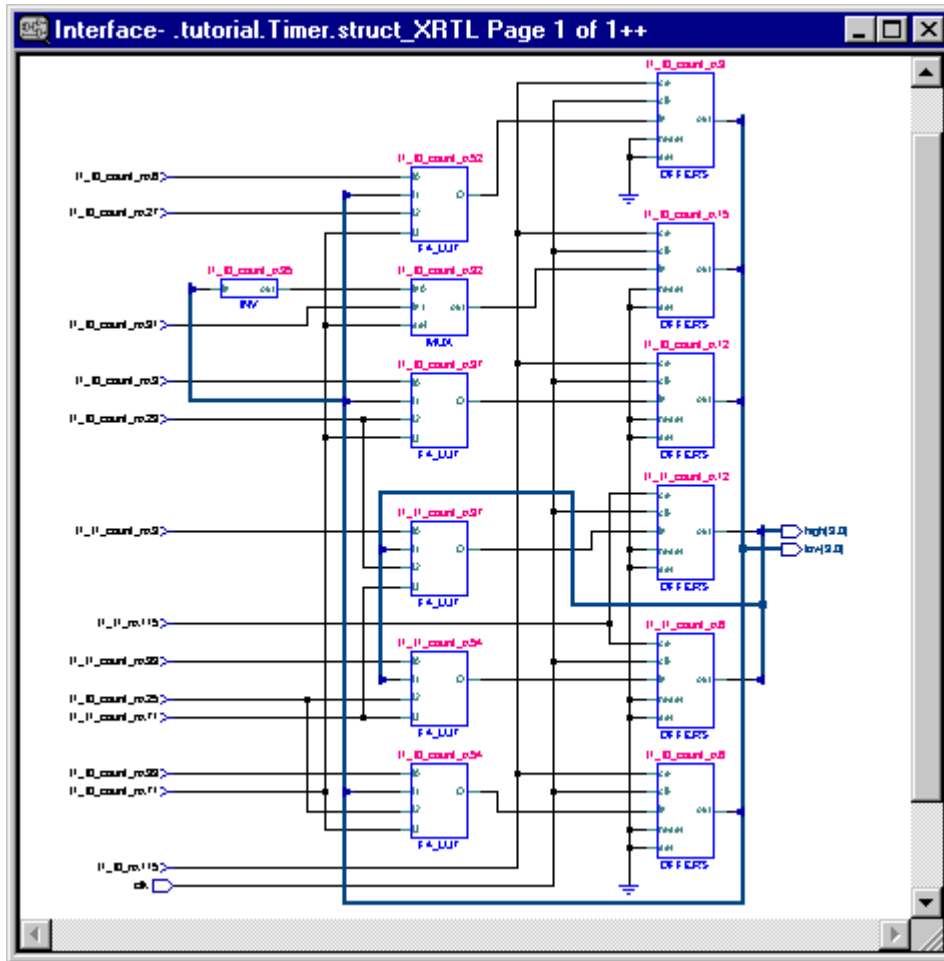
Higher level licenses enable more features but you can choose to run at a lower level using a higher level license.

When you confirm the license, your design is synthesized and optimized for the selected technology.

Status messages are transcribed in the Command window ending with the message “HDL Designer Synthesis run finished”:



You can use the  button from the LeonardoSpectrum toolbar to display the RTL schematic or the  button to display the technology schematic for your design. For example, the following picture shows the RTL schematic:



You can move around the schematic using the scroll bars or the diagram can be enlarged or maximized in the schematic window by choosing the **Zoom In** or **Zoom Fit** options from the **Zoom** cascade of the popup menu.

You can crossprobe to the corresponding source design object by selecting an object in the schematic window and choosing **Trace to HDL Designer** from the popup menu.

- Exit from the synthesis tool by choosing **Exit** from the LeonardoSpectrum **File** menu and respond when you are prompted whether to save your project settings.

You have now completed this tutorial. If you have not successfully verified your design, a completed reference example is available in the examples subdirectory of the HDL Designer Series installation. This example can be accessed by opening the *TIMER_Vlog* library in the design explorer.

Using the Example Verilog Design

A completed example design can be accessed from the *TIMER_Vlog* library in the *Examples* project. The example design includes generated HDL but only dummy library mapping for downstream data.

To use the example design, you should change the location of the downstream data directory to a location for which you have write permissions. Refer to Editing Library Mapping in the *HDL Designer Series User Manual* for information about changing library mapping.

Alternatively, you can easily create your own copy of the example design by using the procedure described in Using the Example Designs in the *Start Here Guide for the HDL Designer Series*.



Windows users typically have write access to the installation directories. However, it is recommended that you change the mapping to a suitable directory for user data rather than overwrite the installed examples.

Once you have modified the library mapping, you can browse, compile and simulate the example design by following the procedures from “Browse the Completed Design” on page 2-73.

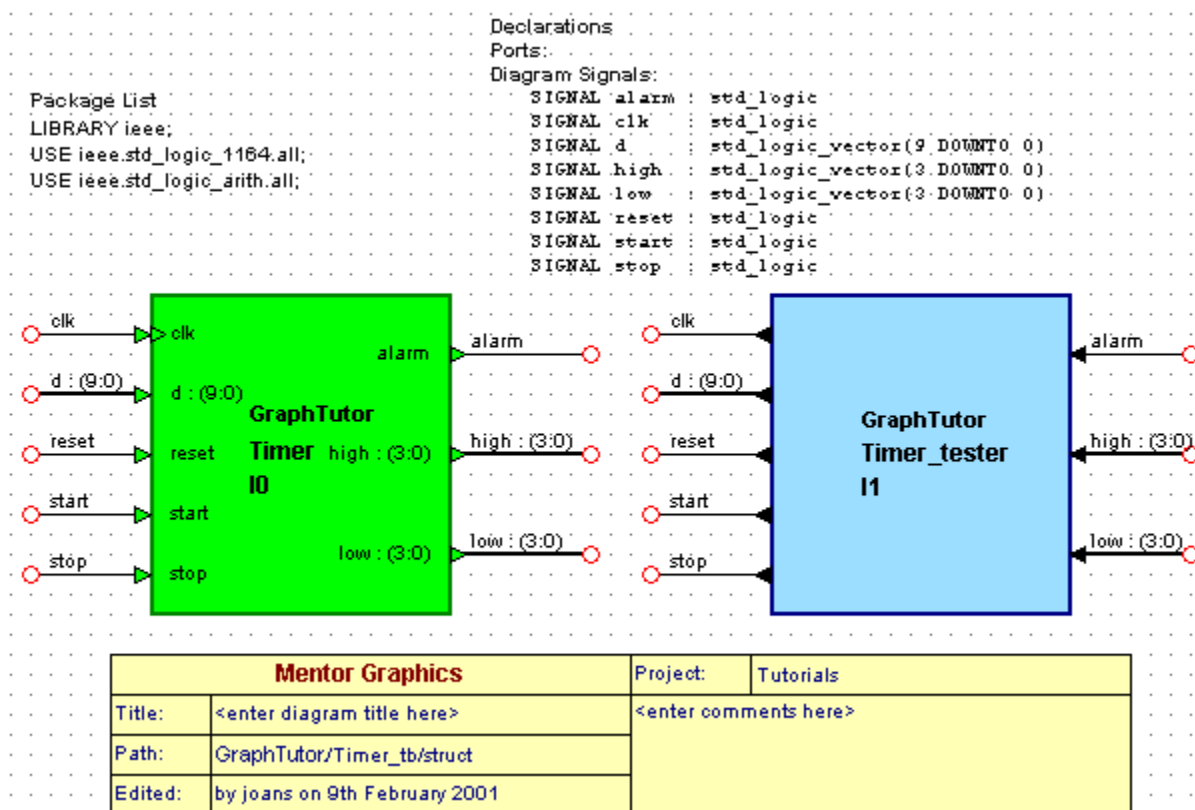
In order to animate the flow chart and state machine in the example design, you must regenerate HDL through components after setting **Instrument HDL for animation** in the state machine and flow chart properties. (Refer to “Set Generation Properties” on page 2-39 and “Generate HDL for the Test Bench” on page 2-71.)

Chapter 3 Creating a VHDL Flow Chart

Introduction

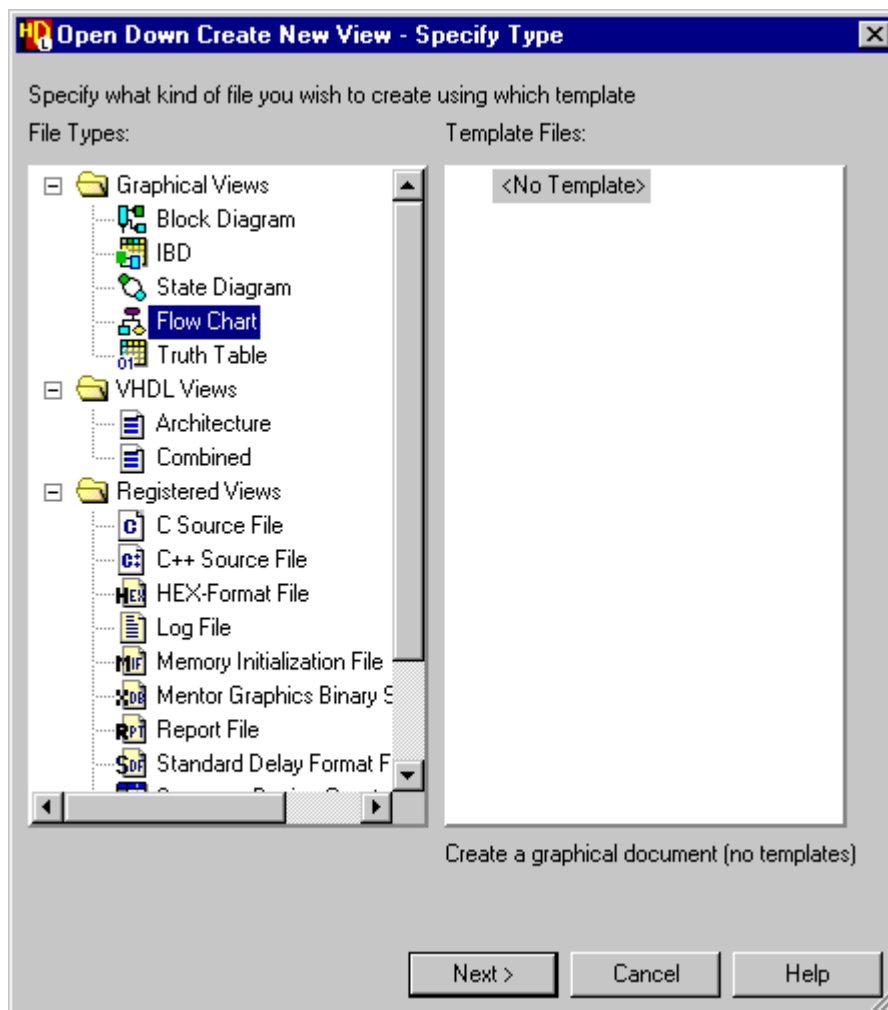
This chapter describes alternative procedures for using a VHDL flow chart instead of importing the *Timer_tester* as a HDL text view.

After you “[Create a Test Bench](#)” on page 1-65 the test bench block diagram should look similar to the following picture:



Create the Tester Flow Chart

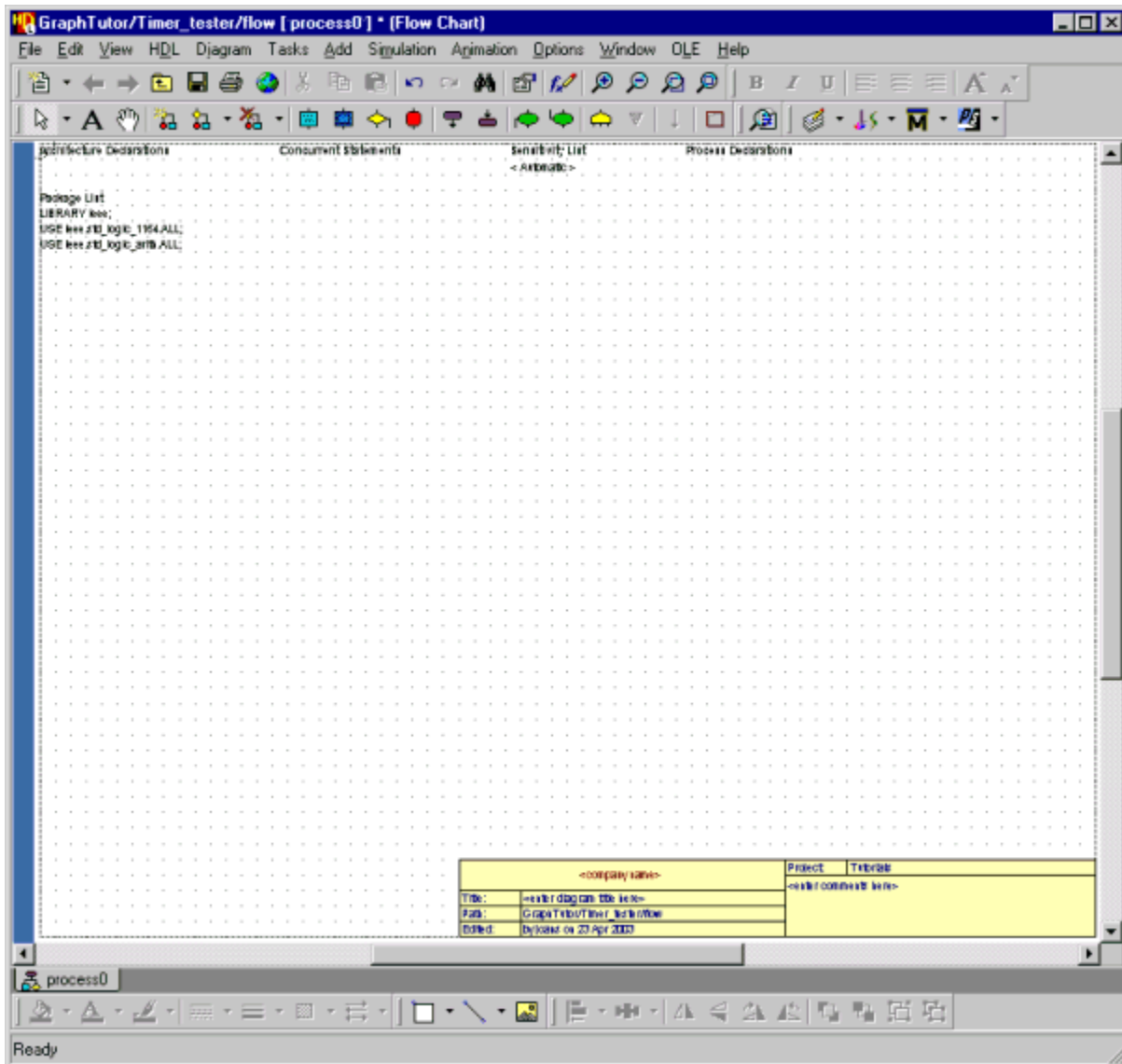
1. Double-click over the *Timer_tester* block in the test bench block diagram and select a **Flow Chart** view in the Open Down Create New View wizard.



2. Use the button to display the Specify View Name page and confirm the wizard using the default view name *flow.fc*.

A new flow chart (*GraphTutor/Timer_tester/flow [process0]*) is created as a child view of the *Timer_tester* block.

The new flow chart is initialized with page boundaries set for the default printer, the default VHDL package list, default title block and labels for architecture declarations, concurrent statements, sensitivity list and process declarations:



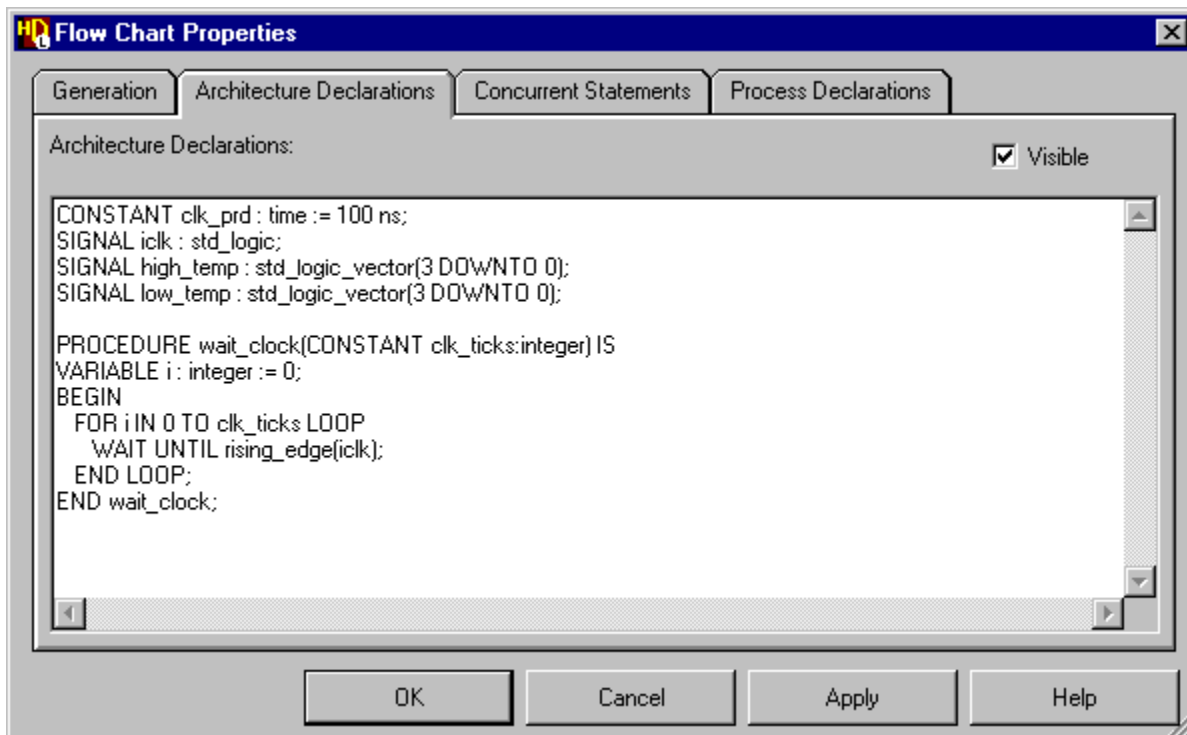
3. Choose **Rename Flow Chart** from the **Diagram** menu and enter the new name *Monitor*. When you confirm the dialog box, this name replaces the default process name *process0* appended to the design unit and view names.

Set Flow Chart Properties

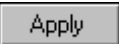
4. Double-click with the mouse over the Architecture Declarations label in the flow chart to display the **Architecture Declarations** tab of the Flow Chart Properties dialog box.
5. Enter the following declarations which set up some internal variables and a simple wait procedure:

```
CONSTANT clk_prd : time := 100 ns;
SIGNAL iclk : std_logic;
SIGNAL high_temp : std_logic_vector(3 DOWNTO 0);
SIGNAL low_temp : std_logic_vector(3 DOWNTO 0);

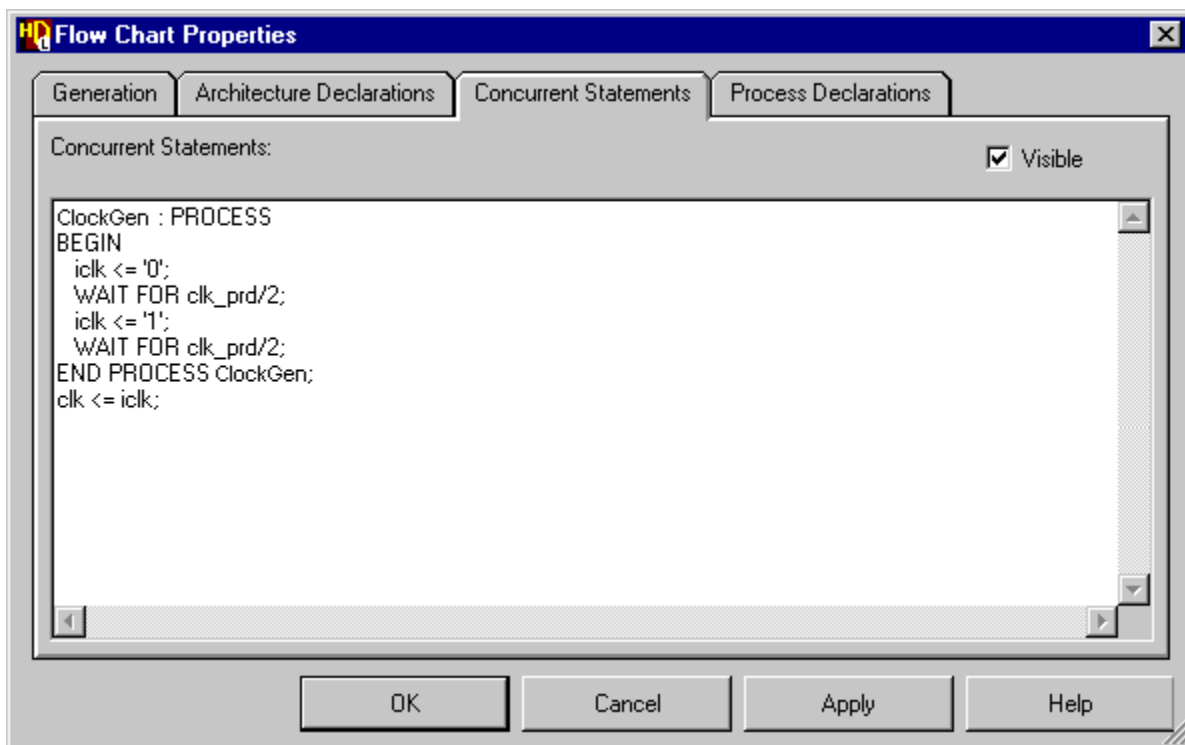
PROCEDURE wait_clock(CONSTANT clk_ticks:integer) IS
VARIABLE i : integer := 0;
BEGIN
    FOR i IN 0 TO clk_ticks LOOP
        WAIT UNTIL rising_edge(icmp);
    END LOOP;
END wait_clock;
```



The architecture declarations are included at the top of the architecture description in the generated HDL and must be valid VHDL statements terminated by a semi-colon. Comment text can be included provided each comment is preceded by the VHDL comment characters (--).

6. Use the  button to display these properties on the flow chart. The HDL syntax is checked on entry and you are warned if any errors are encountered.
7. Select the **Concurrent Statements** tab and enter the following statements which generate a clock signal for the test bench:

```
ClockGen : PROCESS
BEGIN
    iclk <= '0';
    WAIT FOR clk_prd/2;
    iclk <= '1';
    WAIT FOR clk_prd/2;
END PROCESS ClockGen;
clk <= iclk;
```

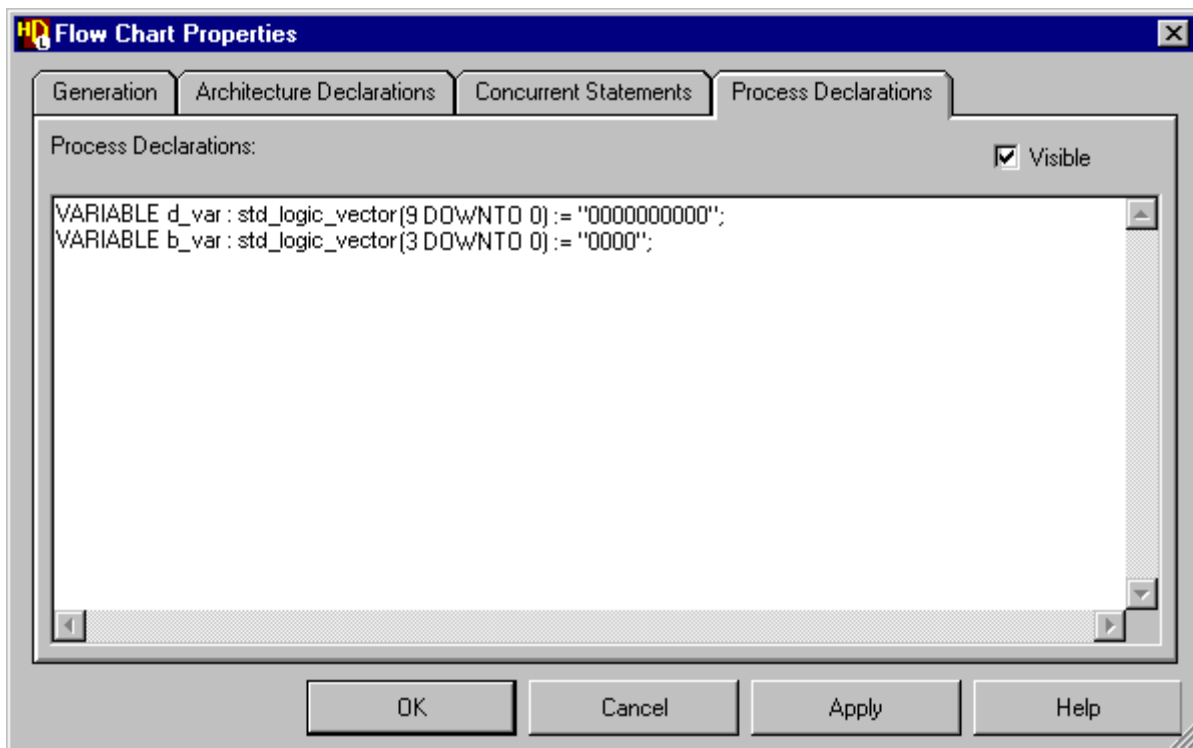


- Use the  button to display these properties on the flow chart.


Concurrent statements are included after the BEGIN statement in the generated HDL for the flow chart and must be valid HDL statements and be terminated by a semi-colon. They are typically used for a concurrent process (such as the clock generator defined by this example).

- Select the **Process Declarations** tab and enter the following variable declarations:




```
VARIABLE d_var : std_logic_vector(9 DOWNTO 0) := "0000000000";  
VARIABLE b_var : std_logic_vector(3 DOWNTO 0) := "0000";
```





Process declarations are included at the top of the main process in the generated HDL for the flow chart and must be valid HDL statements terminated by a semi-colon.

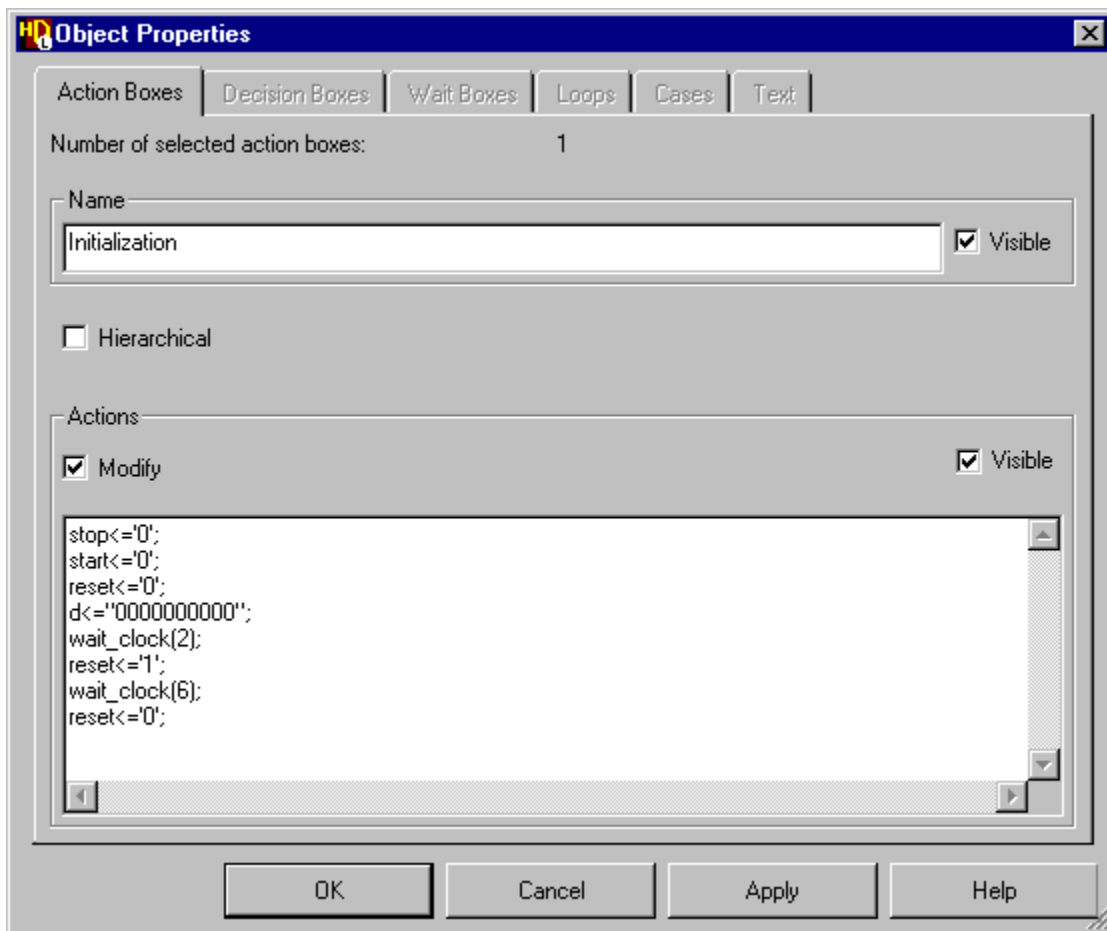
- Use the  button to apply these properties to the flow chart and dismiss the dialog box.

Add a Start Point and Action Box

11. Use the  button to add a start point near the top of the flow chart.
12. Use the  button to add an action box below the start point.
13. Use the  button to add a flow connecting it to the start point.

To add a flow, click with the mouse over the body of the start point and the action box close to the  markers.


14. Double-click with the mouse over the action box (or use the  button) to display the **Action Boxes** tab of the Flow Chart Object Properties dialog box.

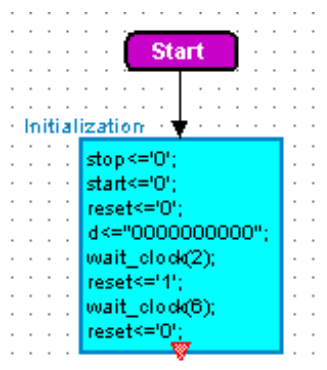


You can change the action box name and enter action statements. However, the name must be unique on the flow chart and cannot be changed when more than one action box is selected.

- Change the default name in the dialog box to *Initialization* and enter the following actions:


```
stop<='0';
start<='0';
reset<='0';
d<="0000000000";
wait_clock(2);
reset<='1';
wait_clock(6);
reset<='0';
```

- Use the  button to update the flow chart. You may want to resize the action box on the chart (by dragging the handles on the lower edge with the mouse) so that it encloses the actions text.



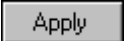
If you drag one side of an action box, the mid-point is preserved.

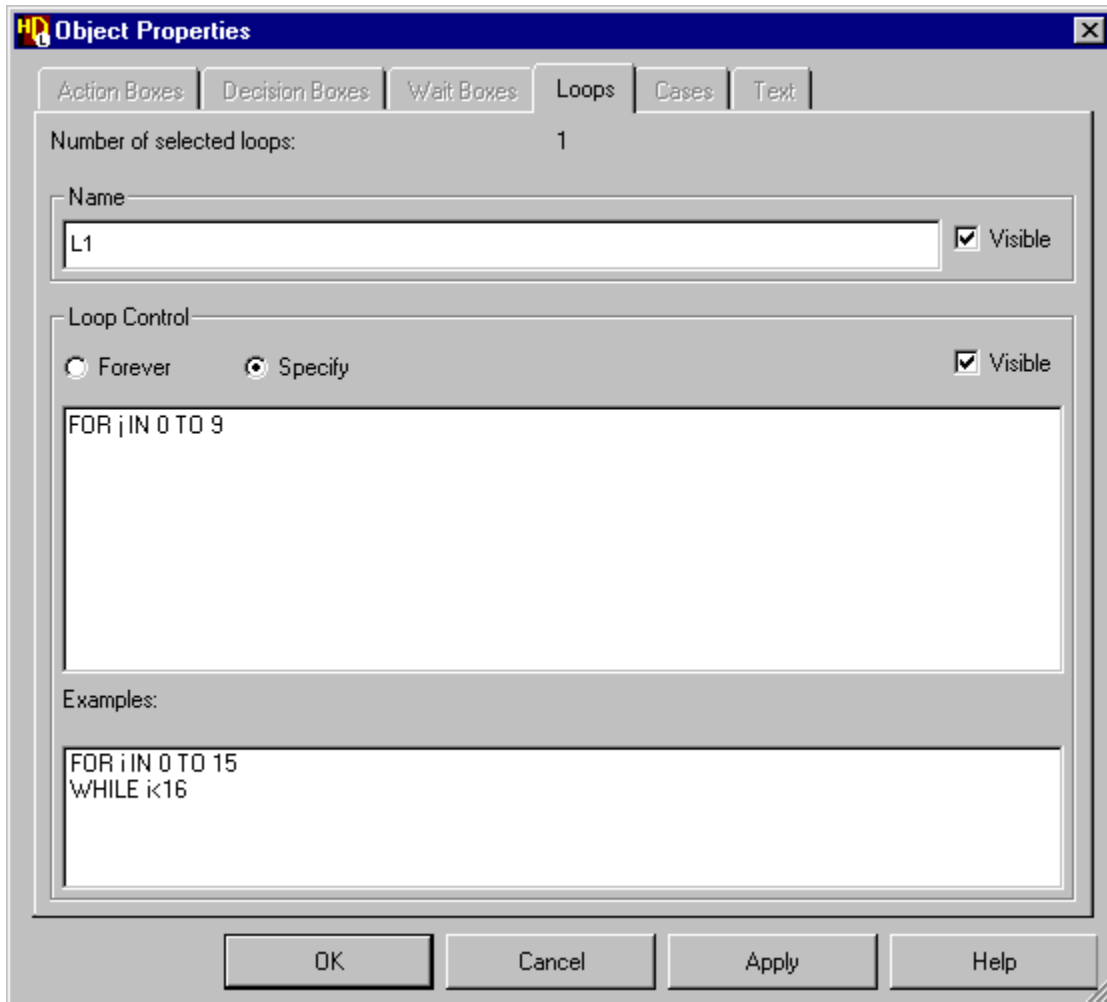
Add a Loop and an Associated Comment


- Use the  button to add a start 1 below the *Initialization* action box.


18. Double-click the start loop to display the **Loops** tab of the Flow Chart Object Properties dialog box. Choose the **Specify** button and enter the following loop control statement:

```
FOR j IN 0 TO 9
```

19. Rename the loop *L1* and use the  button to update the flow chart.







-  By default, a loop will be repeated forever. However, if you choose the **Specify** option, the dialog box discloses an entry box for loop control statements. Template examples are provided which you can copy into the entry box by clicking on one of the examples with the mouse.


20. Use the  button to add an action box below the start loop.

21. Click with the mouse to select the new action box and use the **Action Boxes** tab of the Flow Chart Object Properties dialog box to enter the following actions:

```
d_var:="0000000000";
d_var(j):='1';
d<=d_var;
wait_clock(4);
ASSERT ((high=b_var)AND(low=b_var))
REPORT "Decoder or Load failure."
SEVERITY failure;
b_var:=unsigned(b_var)+1;
```

22. Change the name of the action box to *Decoder* and use the  button to update the flow chart. Resize the action box so that it encloses the actions text.
23. Use the  button to add an end loop below the *Decoder* action box and use the  button to connect flows for the loop.
24. Use the  button to enter text mode. Click in a free space near the loop and enter the following text in the comment text entry box:

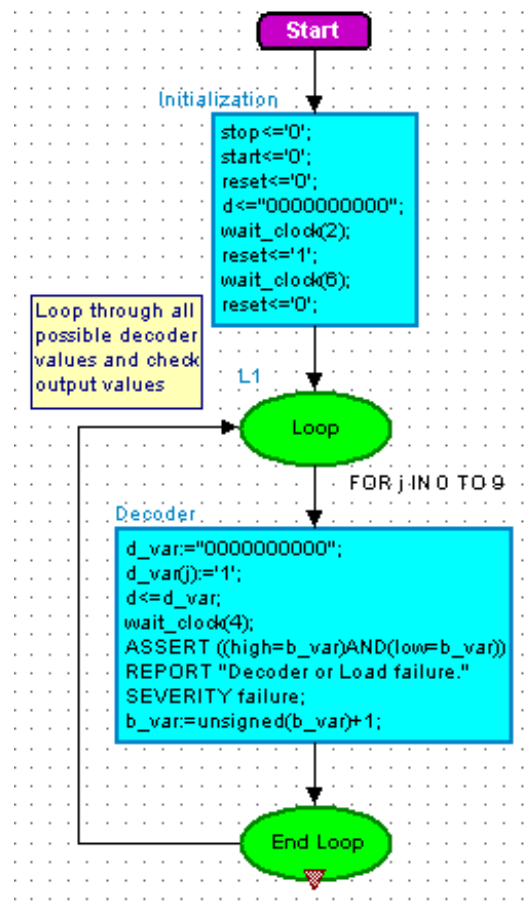
```
Loop through all
possible decoder
values and check
output values
```

-  You can enter free-format text including line breaks and spaces which will be preserved on the diagram.


25. Click the left mouse button outside the text entry box to complete the text.
26. Select the comment text and choose **Include in HDL** from the popup menu. Select **Before Object** from the popup menu. An anchor is attached to the cursor. Click the left mouse button with the cursor over the start loop object to terminate the anchor and associate the notes with the loop.

The comment text will be included immediately before the loop statements in the generated HDL for the flow chart.

The flow chart should look similar to the following picture:



Add an Action Box

27. Use the  button to add another action box below the end loop and use the Flow Chart Object Properties dialog box to rename the action box *Store* and apply the following actions:

```

d<="0000001000";
wait_clock(4);
start<='1';
wait_clock(4);
start<='0';
d<="0000000000";
wait_clock(8);
  
```


```

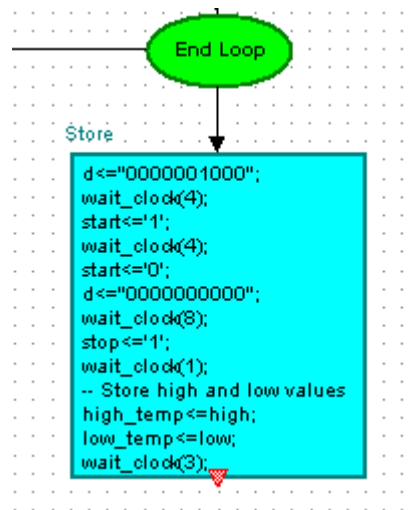
stop<='1';
wait_clock(1);
--Store high and low values
high_temp<=high;
low_temp<=low;
wait_clock(3);

```



You can include comment text directly in an actions box by using HDL comment characters. For example the comment text `--Store high and low values` in the example above.

28. Use the  button to connect flows between the end loop and action boxes.




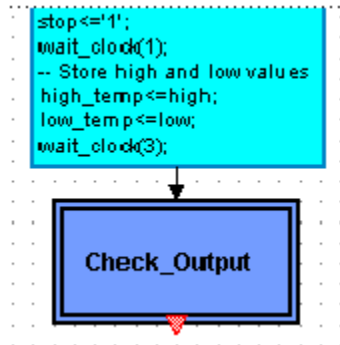
Add a Hierarchical Action Box


When a flow chart contains a large number of procedural statements it can rapidly become a very large diagram which can be difficult to read or print. However, such a chart can be represented as a number of separate hierarchical charts by using hierarchical action boxes.



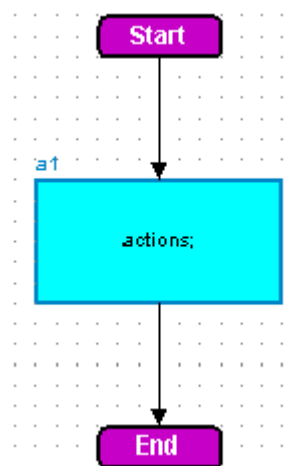
Flow chart hierarchy does not effect the generated HDL and is not used if you use HDL import to recover the *Timer_tester* flow chart.

29. Use the  button to add a hierarchical action box below the *Store* action box and use direct text editing to change the name of the hierarchical action box to *Check_Output*.




30. Use the  button to connect a flow between the action box and hierarchical action box.
31. Double-click on the *Check_Output* hierarchical action box to open a new child flow chart (or select the hierarchical action box and choose **Open Down** from the **File** menu).


The child flow chart is initialized in a new window tab as an action box connected by flows to a start point and end point.



Add a Decision Box

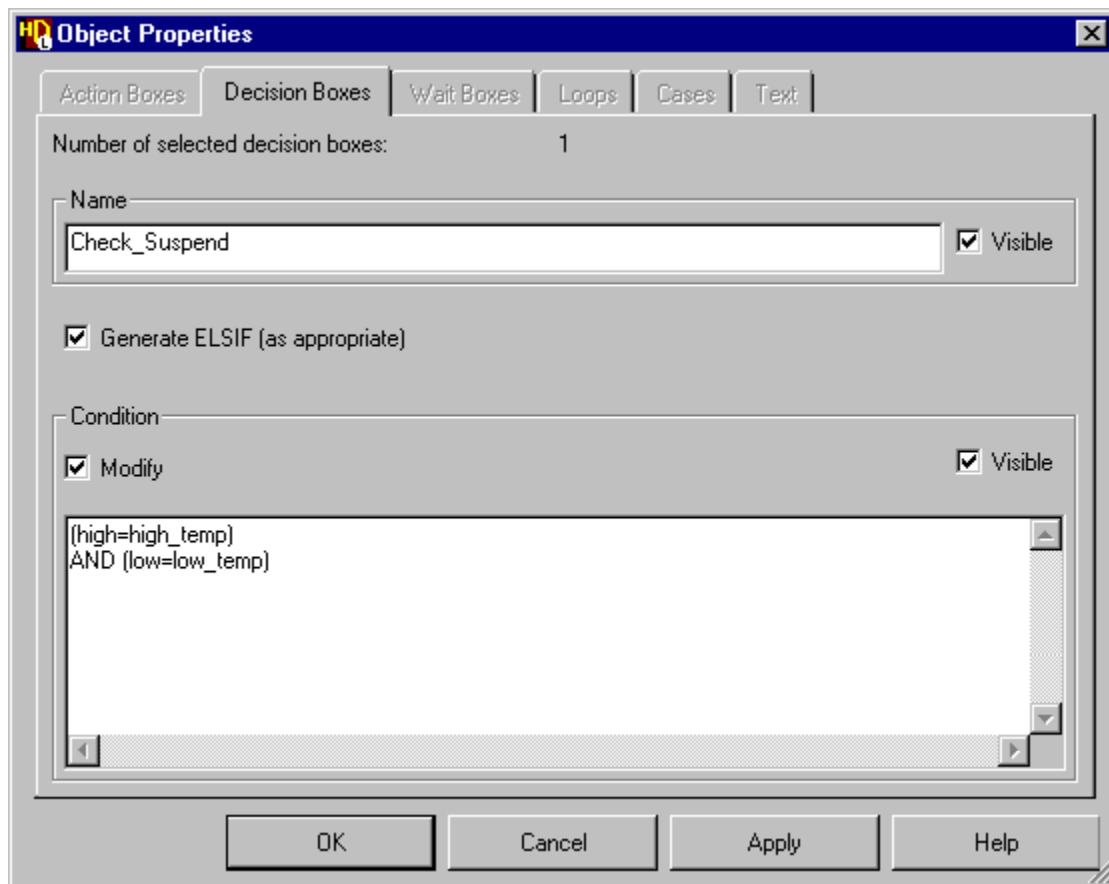
32. Select the flows connected to the start point and end point in the child flow chart and use the **Del** key to delete them.

 You can use **Shift** key with the left mouse button to select both flows.

33. Use the  button to add a decision box below the start point in the *Check_Output* flow chart.

34. Use the **Decision Boxes** tab of the Flow Chart Object Properties dialog box to change the name of the decision box to *Check_Suspend* and enter the following condition:


```
(high=high_temp)
AND (low=low_temp)
```



35. Select the existing action box. Use the **Action Boxes** tab of the Flow Chart Object Properties dialog box to change its name to *Pass* and enter the following actions:


Pass

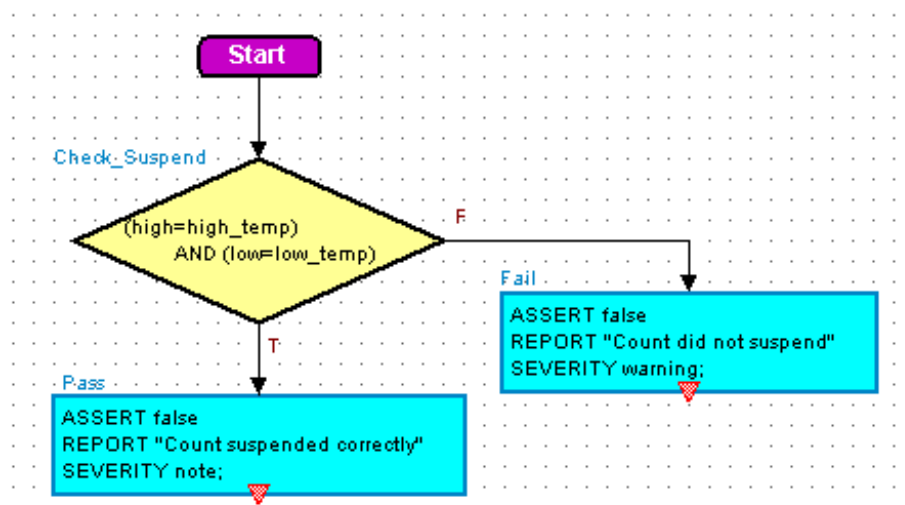
```
ASSERT false
REPORT "Count suspended correctly"
SEVERITY note;
```

36. Use the  button to add an action box for the False (F) condition. Use the **Action Boxes** tab to change its name to *Fail* and enter the following actions:

Fail


```
ASSERT false
REPORT "Count did not suspend"
SEVERITY warning;
```

37. Reposition the action boxes by dragging with the mouse and use the  button to connect flows between the start point, decision box and action boxes.





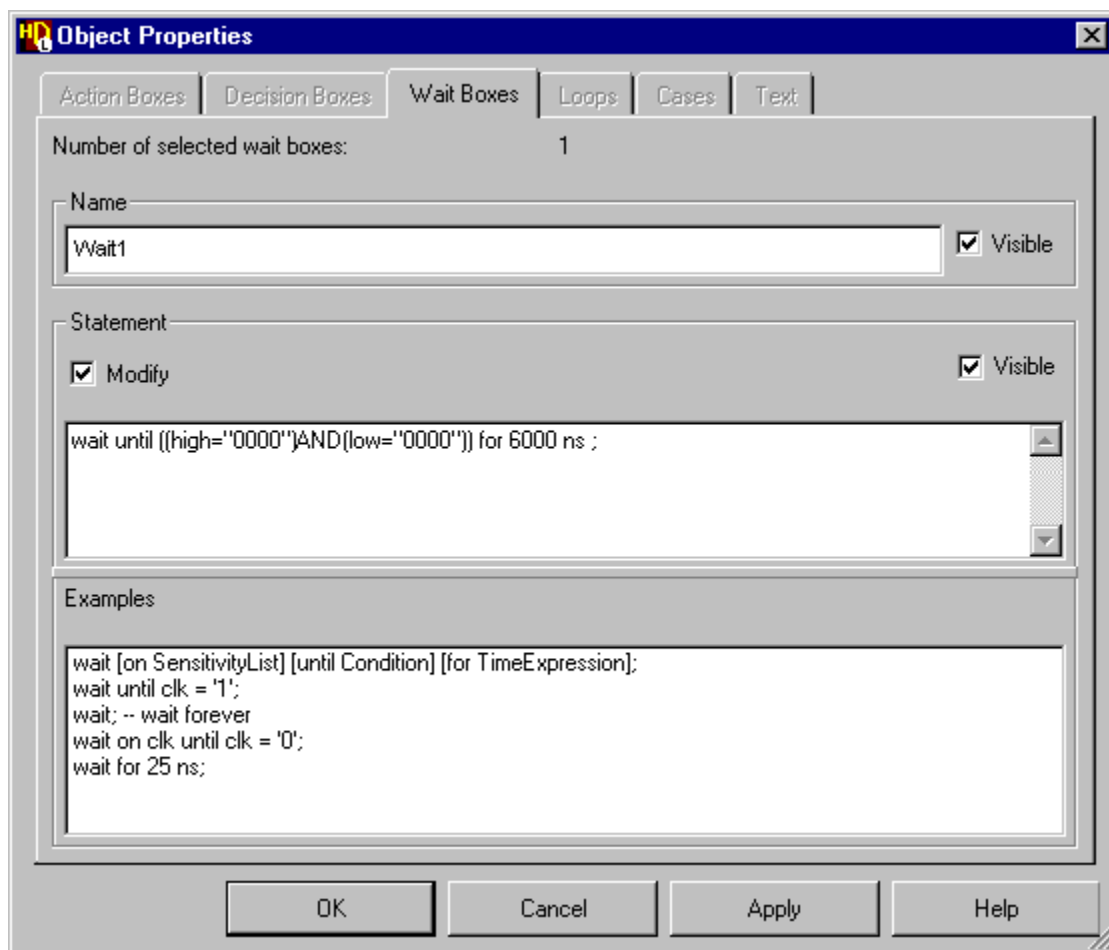
When a decision box object is selected, the popup menu includes an option to swap the True and False outputs.

Add Wait Boxes

38. Use the  button to add an action box below the *Pass* action box and use the **Action Boxes** tab of the Flow Chart Object Properties dialog box to enter the following actions:

```
wait_clock(4);
stop<='0';
```

39. Change the name of this action box to *Continue* and use the  button to add two wait boxes below it.
40. Click the mouse over the first wait box (or use the  button) to display the **Wait Boxes** tab of the Flow Chart Object Properties dialog box:



- Use the **Wait Box** tab of the Flow Chart Object Properties dialog box to rename the wait boxes and specify the following wait conditions:

Wait1

```
wait until ((high="0000")AND(low="0000")) for 6000 ns ;
```

Wait2

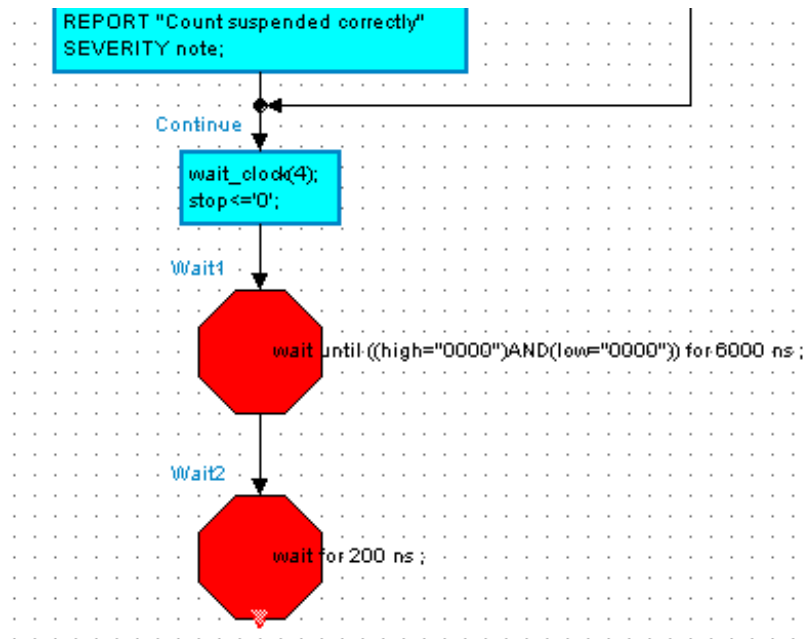
```
wait for 200 ns ;
```



You can click on one of the examples shown in the dialog box to use it as a template. Any valid combination of **on**, **until** and **for** can be used in the wait statement which must be terminated by a semi-colon.




- Use the  button to connect flows between the action boxes and wait boxes.

Notice that a flow join is automatically created when you connect the two flows from the *Pass* and *Fail* action boxes together.



Copy the Decision Tree

The child hierarchical flow chart can be completed by using similar procedures to those described in the last two topics. However, the next section can be done more quickly by copying the decision tree objects which have already been created.

43. Select the *Check_Suspend* decision box, plus the *Pass* and *Fail* action boxes and use the  button (or choose **Copy** from the popup or **Edit** menu).
44. Scroll the window down and click the  mouse button below the last wait box to de-select the objects. Use the  button (or choose **Paste** from the popup or **Edit** menu) to paste a copy of the objects from the clipboard in the middle of the window. Use the mouse to drag the objects to the required position. Notice that each of the pasted objects is automatically given a unique name.
45. Use the Flow Chart Object Properties dialog box to change the names, conditions and actions where required.

Check_Alarm


```
alarm='1'
```

Alarm_Pass

```
ASSERT false
REPORT "Alarm asserted correctly"
SEVERITY note;
```

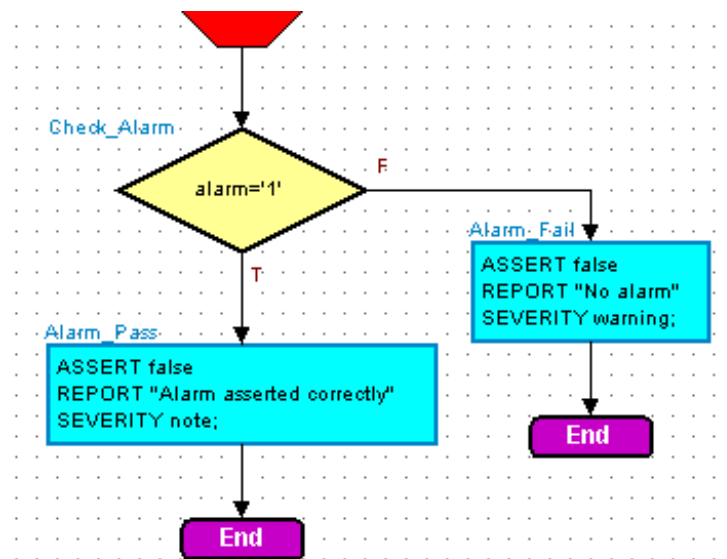
Alarm_Fail

```
ASSERT false
REPORT "No alarm"
SEVERITY warning;
```



46. Drag the end point beneath the other objects and use the  button to connect the remaining flows on the child flow chart.



Multiple end points can be used. For example, in the following picture, a separate end point has been attached to the *Alarm_Pass* and *Alarm_Fail* action boxes.




Completing the Flow Chart


47. Use the  button (or choose **Open Up** from the **File** menu) to display the parent flow chart and the  button to add an action box below the hierarchical action box.
48. Use the Flow Chart Object Properties dialog box to change the name of this action box to *Clear* and enter the following actions:

```



stop<='1';
wait_clock(2);
stop<='0';
wait_clock(4);
ASSERT false
REPORT "Timer test completed"
SEVERITY Failure;

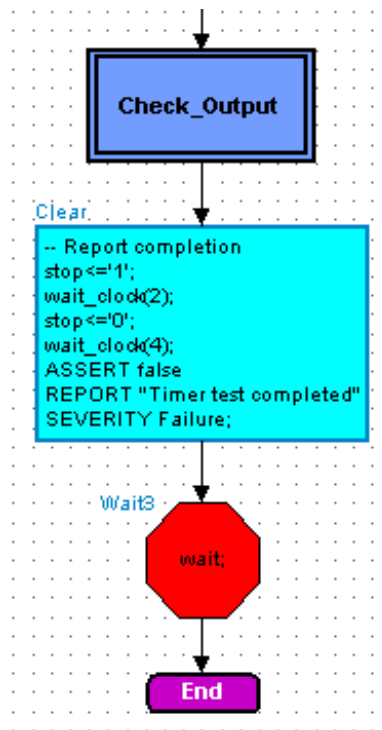
```


-  These actions are executed when the timer test has completed successfully but are given *Failure* severity to stop the simulator execution after the completion message is issued.

49. Use the  button to add a wait box below the action box. Change the name of this wait box to *Wait3* with the default wait condition (*wait;*).

This condition corresponds to a wait for interaction at the completion of the test sequence.

50. Use the  button to add an end point and use the  button to connect any remaining flows.



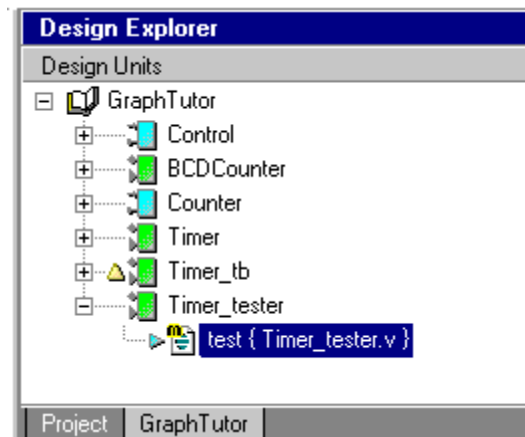
51. Complete the flow chart by editing the title, project and comments fields in the title block.
52. Use the  button to save the flow chart.

Return to [“Generate HDL for the Test Bench”](#) on page 1-70 in the main VHDL tutorial.

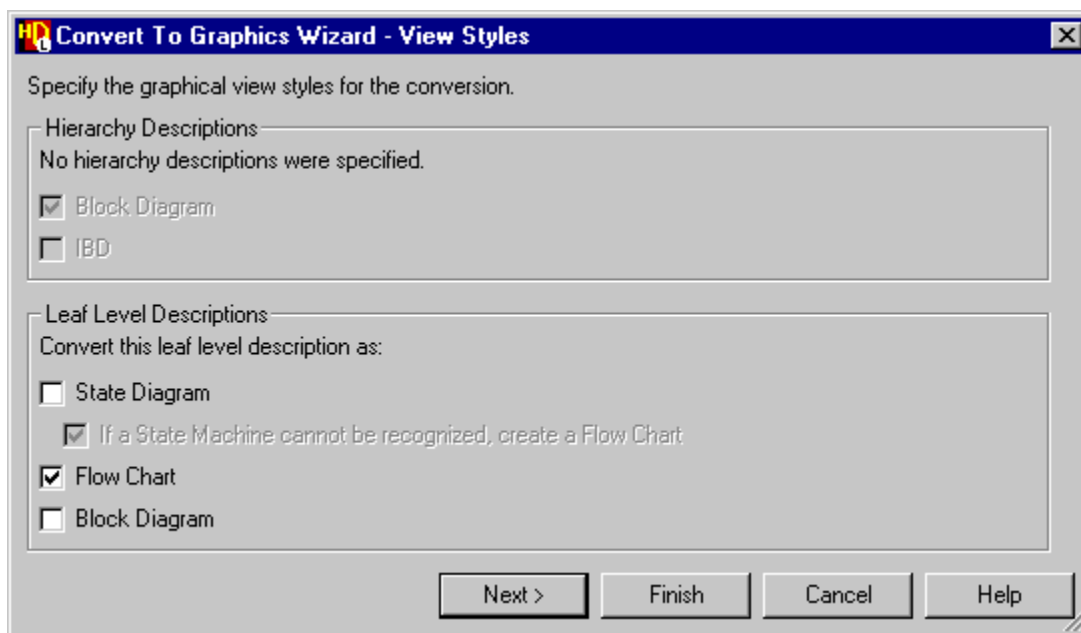
Using HDL2Graphics


If you are using **HDL Designer**, you can use HDL2Graphics to convert the HDL text view of the *Timer_tester* design unit to a graphical flow chart.

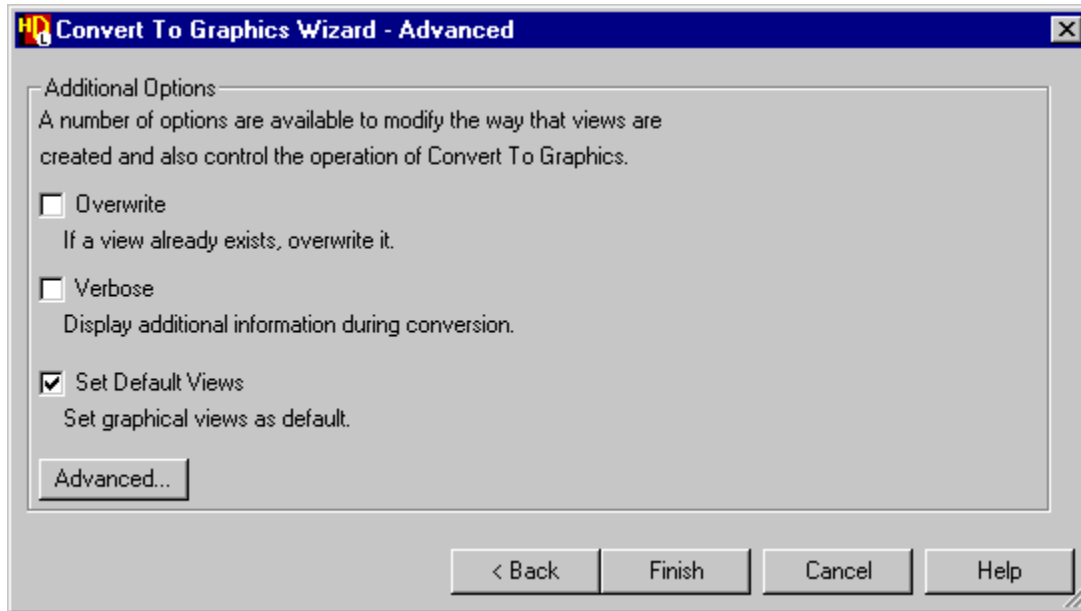
1. Select the Verilog module *test {Timer_tester.v}* for the *Timer_tester* design unit in the *Design Units* view of the design explorer:




2. Use the  button or choose **Single Level** from the **Convert To Graphics** cascade of the **HDL** menu to display the View Styles page of the Convert To Graphics wizard:



3. Set the **Flow Chart** option in the Leaf-level Descriptions section of this page and use the  button to display the Advanced page of the wizard:



4. Set the **Set Default Views** option and use the  button to convert the VHDL architecture view to a flow chart.

You are prompted to confirm that you wish to continue. When you confirm the prompt dialog box, the new graphical view is made the default view but the HDL text view is retained as an alternative view for the design unit.

Notice that a graphical symbol is also created for the *Timer_tester* design unit.

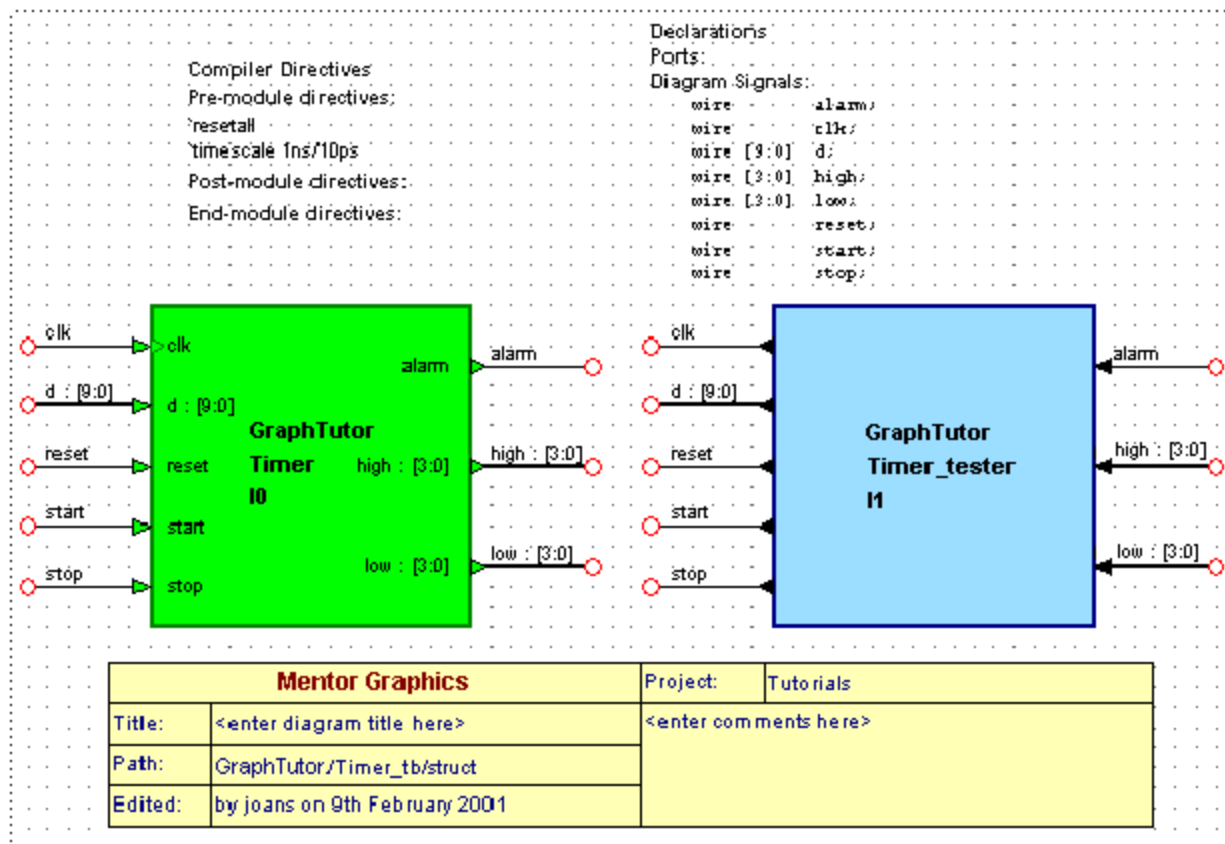
Return to [“Instantiate the Imported Tester” on page 1-68](#) in the main VHDL tutorial.

Chapter 4 Creating a Verilog Flow Chart

Introduction

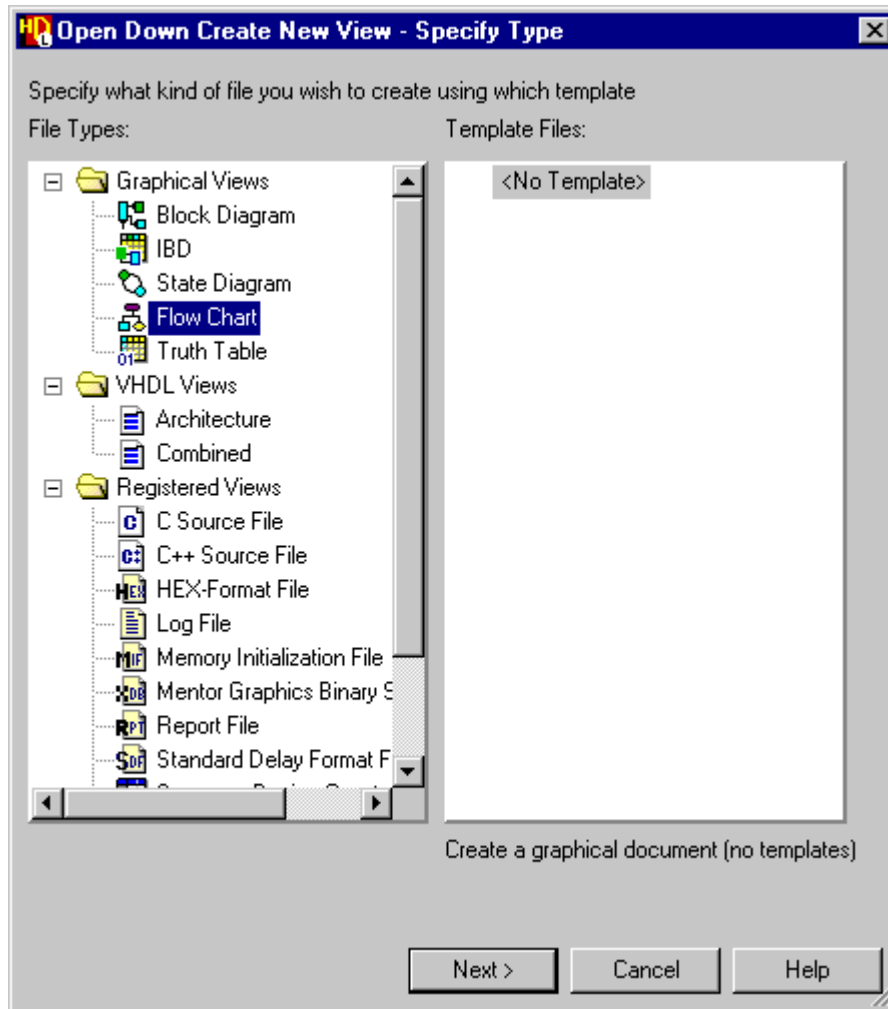
This chapter describes alternative procedures for using a Verilog flow chart instead of importing the *Timer_tester* as a HDL text view.

After you “[Create a Test Bench](#)” on page 2-66 the test bench block diagram should look similar to the following picture:



Create the Tester Flow Chart

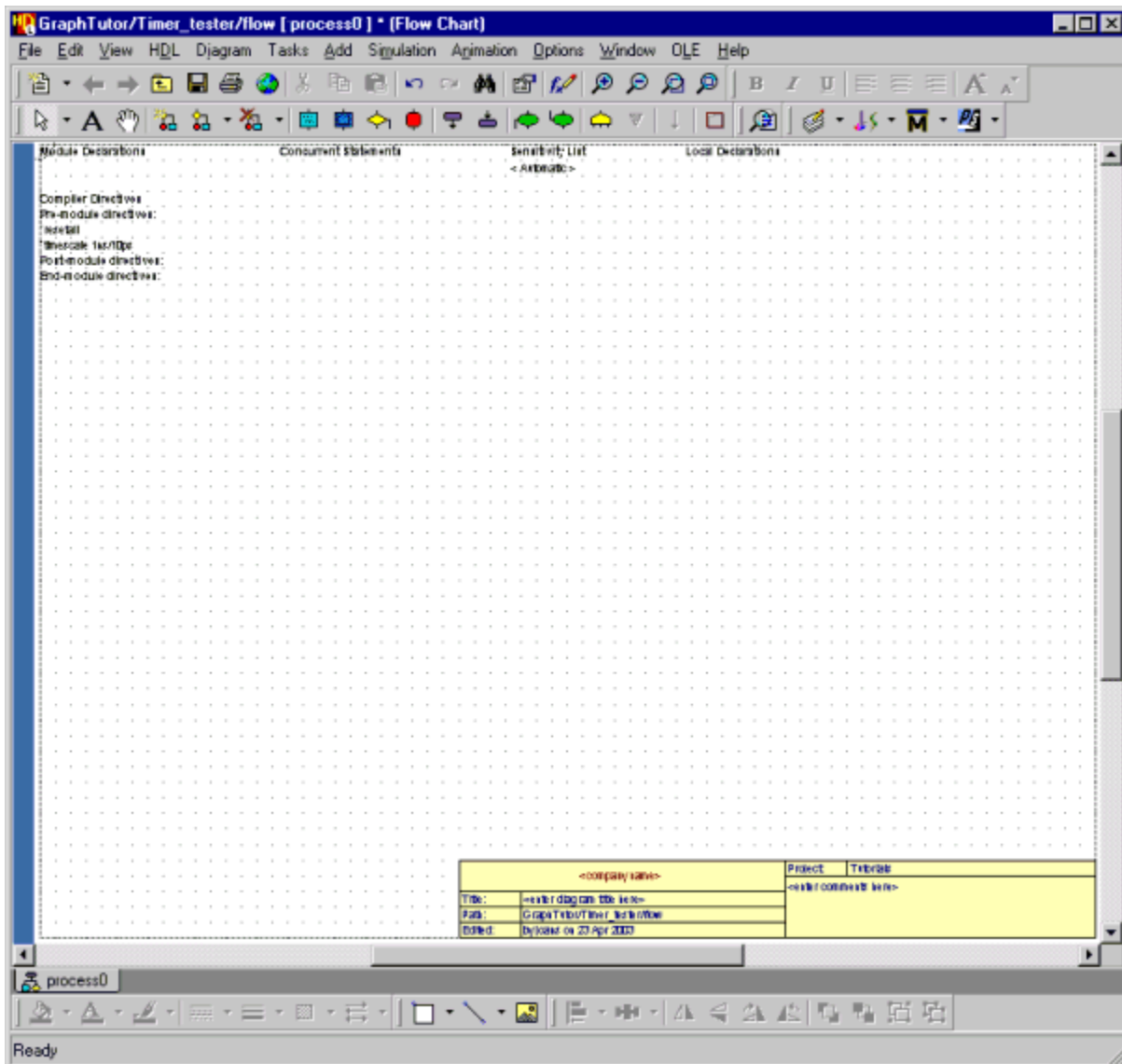
1. Double-click over the *Timer_tester* block in the test bench block diagram and select a **Flow Chart** view in the Open Down Create New View wizard.



2. Use the button to display the Specify View Name page and confirm the wizard using the default view name *flow.fc*.

A new flow chart (*GraphTutor/Timer_tester/flow [process0]*) is created as a child view of the *Timer_tester* block.

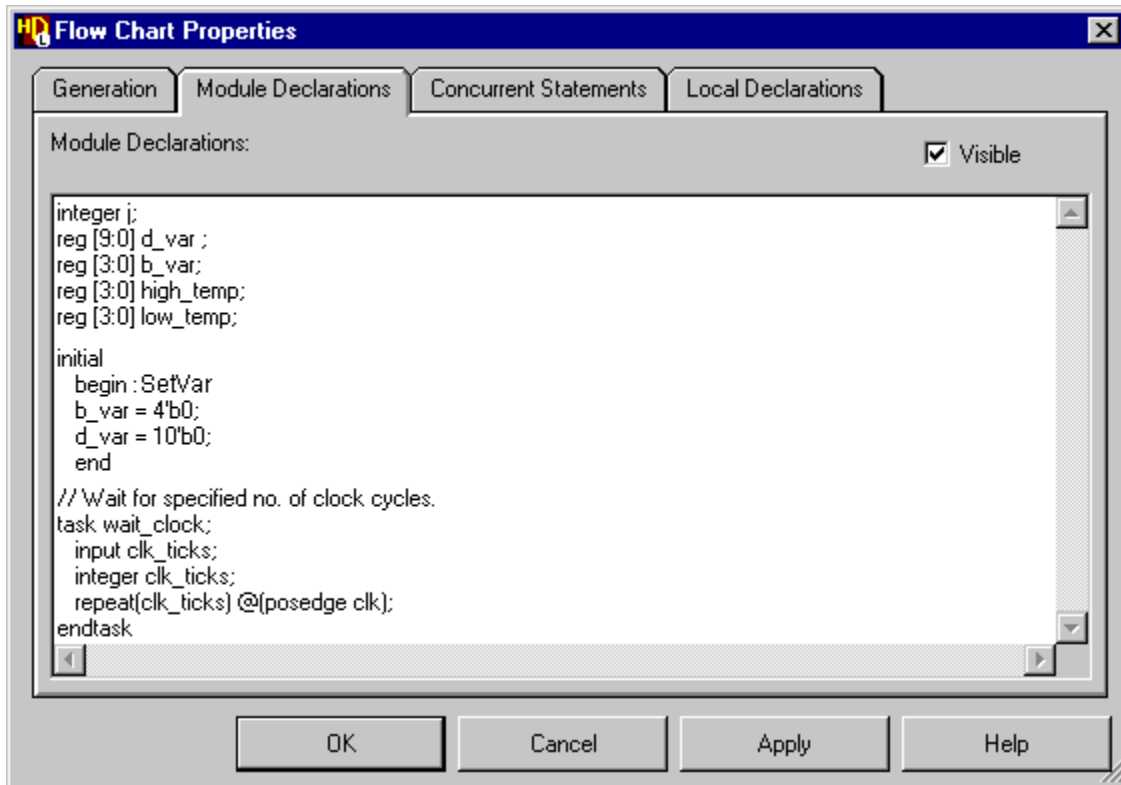
The new flow chart is initialized with page boundaries set for the default printer, the default VHDL package list, default title block and labels for architecture declarations, concurrent statements, sensitivity list and process declarations:



3. Choose **Rename Flow Chart** from the **Diagram** menu and enter the new name *Monitor*. When you confirm the dialog box, this name replaces the default process name (*process0*) appended to the design unit and view names.

Set Flow Chart Properties

1. Double-click with the mouse over the Module Declarations label in the flow chart to display the **Module Declarations** tab of the Flow Chart Properties dialog box.



2. Enter the following declarations which set up some internal variables and a simple wait procedure:

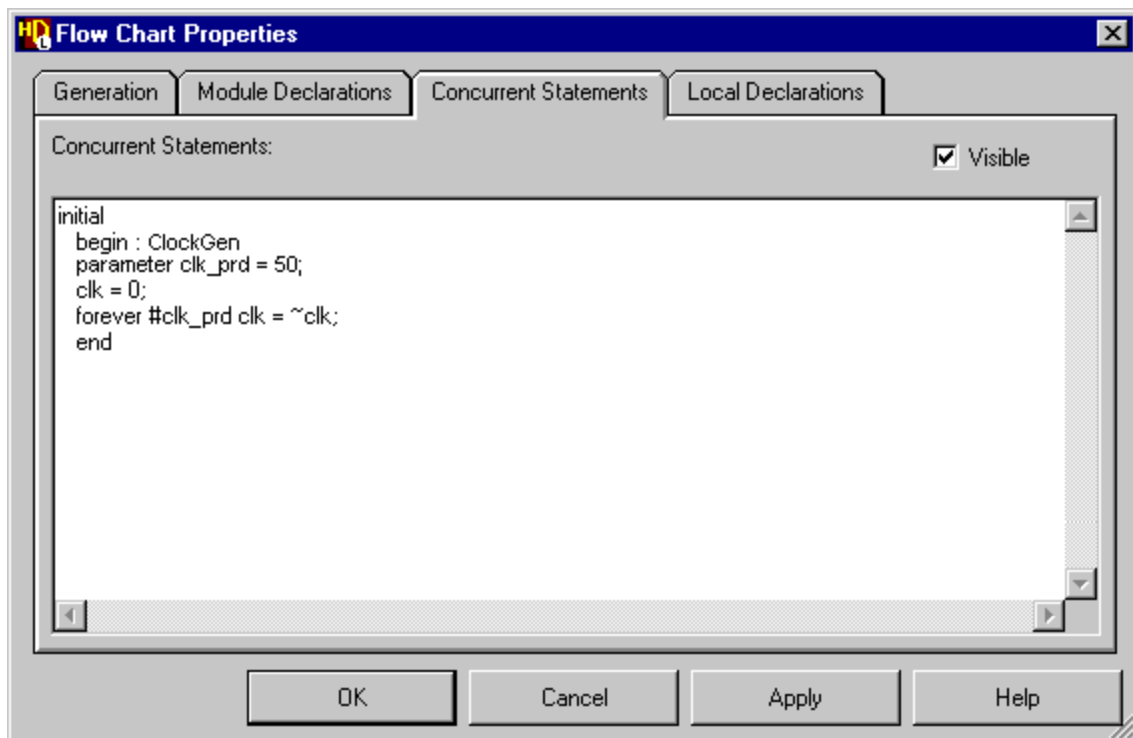
```
integer j;
reg [9:0] d_var ;
reg [3:0] b_var;
reg [3:0] high_temp;
reg [3:0] low_temp;

initial
begin : SetVar
b_var = 4'b0;
d_var = 10'b0;
end
```

```
// Wait for specified no. of clock cycles.  
task wait_clock;  
    input clk_ticks;  
    integer clk_ticks;  
    repeat(clk_ticks) @(posedge clk);  
endtask
```

The module declarations are included at the top of the module description in the generated HDL and must be valid HDL statements terminated by a semi-colon. Comments can also be included provided each comment is preceded by the appropriate Verilog comment characters (//).

3. Use the button to display these properties on the flow chart. The HDL syntax is checked on entry and you are warned if any errors are encountered.
4. Select the **Concurrent Statements** tab.



5. Enter the following statements which generate a clock signal for the test bench:

```
initial
begin : ClockGen
parameter clk_prd = 50;
clk = 0;
forever #clk_prd clk = ~clk;
end
```

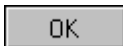
6. Use the  button to display these properties on the flow chart.

Concurrent statements are included between the *end* and *endmodule* statements in the generated HDL for the flow chart and must be valid HDL statements and be terminated by a semi-colon. They are typically used for a concurrent process (such as the clock generator defined by this example).




7. There are no local declarations used in this design. Select the **Local Declarations** tab and uncheck the **Visible** option in order to hide this label on the chart.





Local declarations are included at the top of the initial or always code in the generated HDL for the flow chart and if entered must be valid HDL statements terminated by a semi-colon.

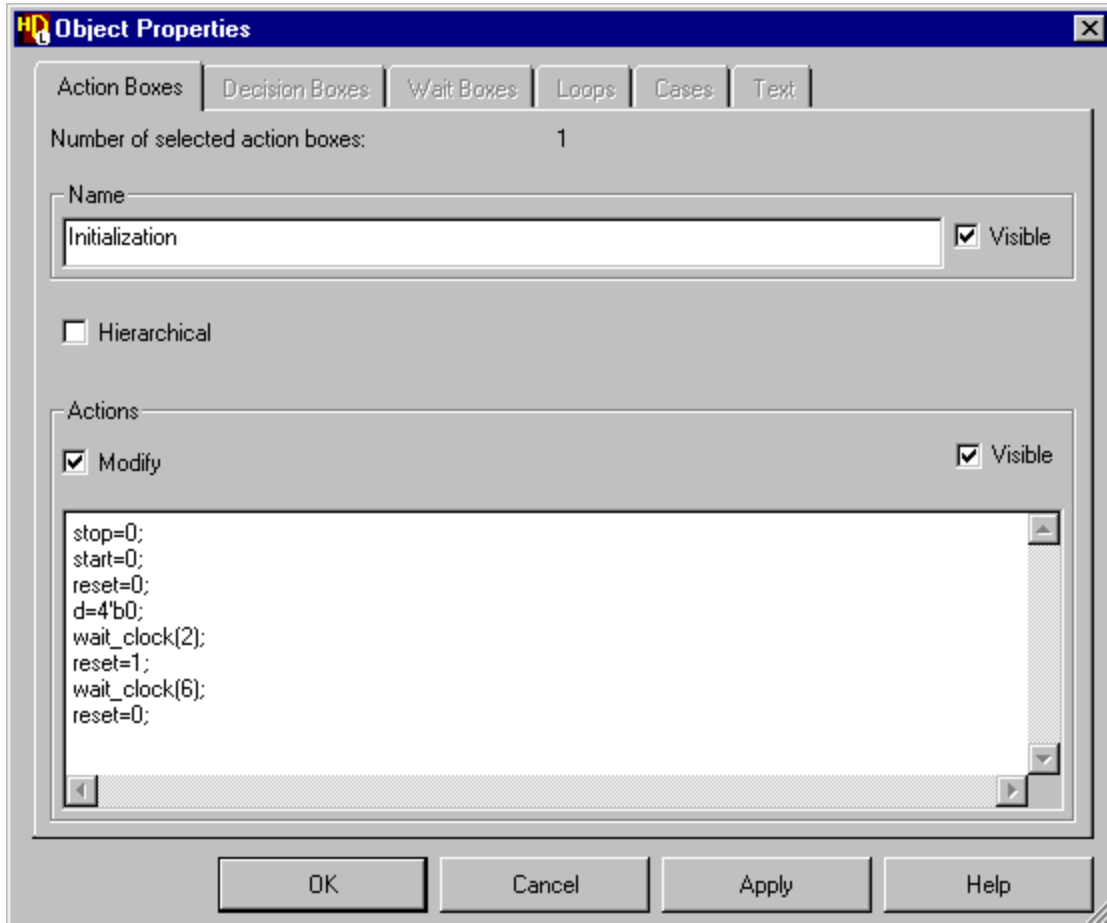
8. Use the  button to confirm and dismiss the Flow Chart Properties dialog box.

Add a Start Point and Action Box

9. Use the  button to add a start point near the top of the flow chart.
10. Use the  button to add an action box below the start point and
11. use the  button to add a flow connecting it to the start point.

To add a flow, click with the mouse over the body of the start point and the action box close to the  markers.

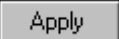
12. Double-click with the mouse over the action box (or use the  button) to display the **Action Boxes** tab of the Flow Chart Object Properties dialog box.

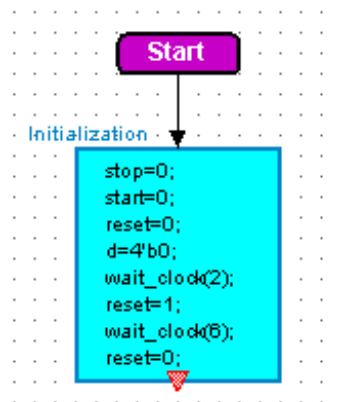


You can change the action box name and enter action statements. However, the name must be unique on the flow chart and cannot be changed when more than one action box is selected.

13. Change the default name to *Initialization* and enter the following actions:


```
stop=0;
start=0;
reset=0;
d=4'b0;
wait_clock(2);
reset=1;
wait_clock(6);
reset=0;
```

14. Use the  button to update the flow chart. You may want to resize the action box on the chart (by dragging the handles on the lower edge with the mouse) so that it encloses the actions text.



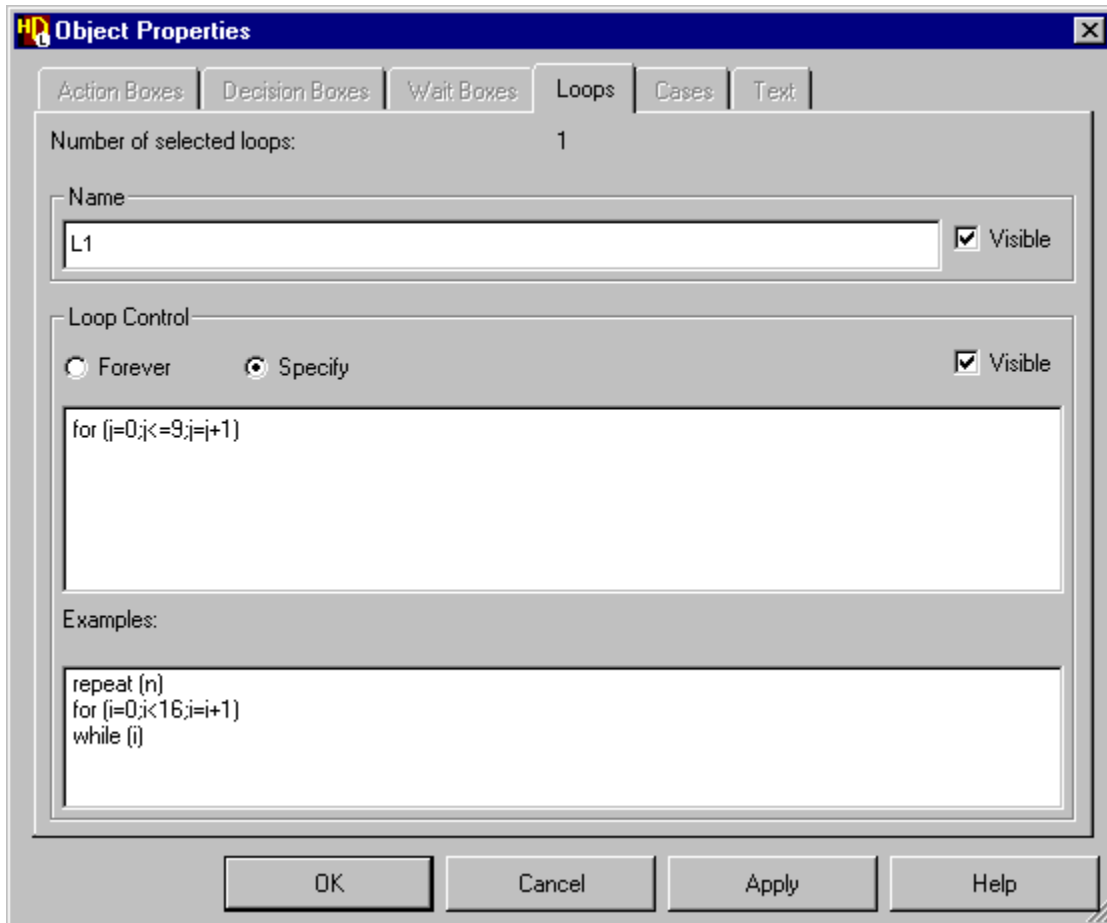
If you drag one side of an action box, the mid-point is preserved.


Add a Loop and an Associated Comment


15. Use the  button to add a start loop below the *Initialization* action box.
16. Double-click the start loop to display the **Loops** tab of the Flow Chart Object Properties dialog box. Choose the **Specify** button and enter the following loop control statement:

```
for (j=0; j<=9; j=j+1)
```

17. Rename the loop *L1* and use the  button to update the flow chart.







-  By default, a loop will be repeated forever. However, if you choose the **Specify** option, the dialog box discloses an entry box for loop control statements. Template examples are provided which you can copy into the entry box by clicking on one of the examples with the mouse.

18. Use the  button to add an action box below the start loop.

19. Click with the mouse to select the new action box and use the **Action Boxes** tab of the Flow Chart Object Properties dialog box to enter the following actions:

```
d_var=10'b0;
d_var[j]=1;
d=d_var;
wait_clock(4);
if ((high!=b_var)&&(low!=b_var))
$display ("Decoder or Load failure.");
b_var=b_var+1'b1;
```

20. Change the name of the action box to *Decoder* and use the  button to update the flow chart. Resize the action box so that it encloses the actions text.
21. Use the  button to add an end loop below the *Decoder* action box and use the  button to connect flows for the loop.
22. Use the  button to enter text mode. Click in a free space near the loop and enter the following text in the comment text entry box:

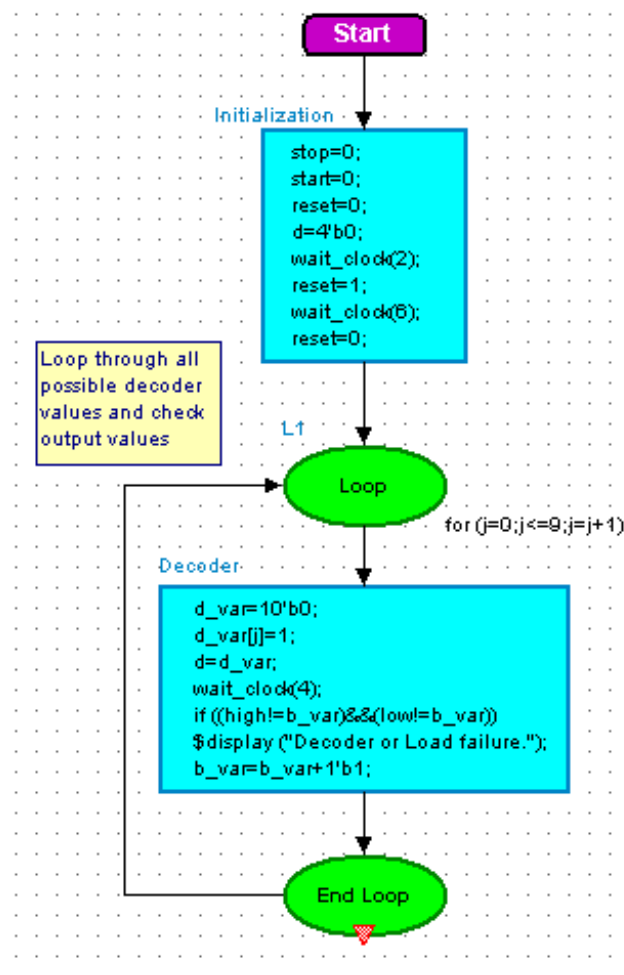
```
Loop through all
possible decoder
values and check
output values
```

-  You can enter free-format text including line breaks and spaces which will be preserved on the diagram.


23. Click the left mouse button outside the text entry box to complete the text.
24. Select the comment text and choose **Include in HDL** from the popup menu. Select **Before Object** from the popup menu. An anchor is attached to the cursor. Click the left mouse button with the cursor over the start loop object to terminate the anchor and associate the notes with the loop.

The comment text will be included immediately before the loop statements in the generated HDL for the flow chart.

The flow chart should look similar to the following picture:



Add an Action Box

- Use the  button to add another action box below the end loop and use the Flow Chart Object Properties dialog box to rename the action box *Store* and apply the following actions:

```


d=10'b0000001000;
wait_clock(4);
start=1;
wait_clock(4);
start=0;
d=10'b0;
    
```

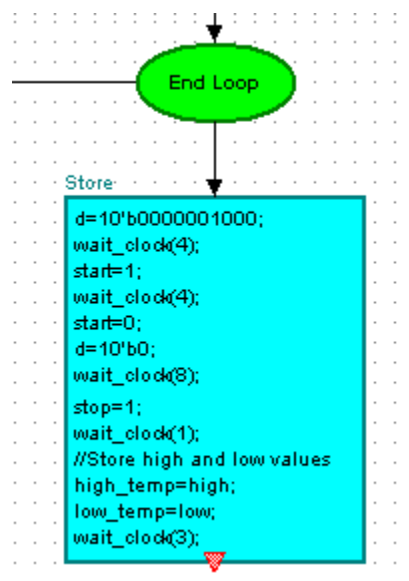
```

wait_clock(8);
stop=1;
wait_clock(1);
//Store high and low values
high_temp=high;
low_temp=low;
wait_clock(3);

```

- i** You can include comment text directly in an actions box by using HDL comment characters. For example the comment text: `//Store high and low values` in the example above.


26. Use the  button to connect flows between the end loop and action boxes.

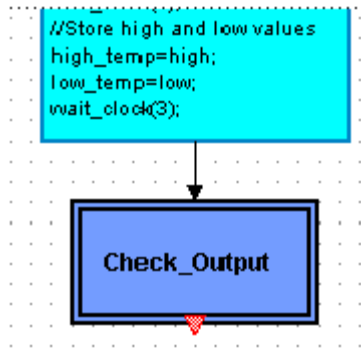



Add a Hierarchical Action Box

When a flow chart contains a large number of procedural statements it can rapidly become a very large diagram which can be difficult to read or print. However, such a chart can be represented as a number of separate hierarchical charts by using hierarchical action boxes.

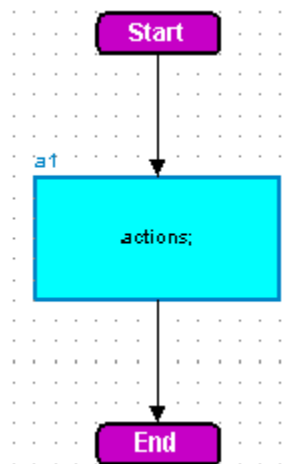
- i** Flow chart hierarchy does not effect the generated HDL and is not used if you use HDL import to recover the *Timer_tester* flow chart.

27. Use the  button to add a hierarchical action box below the *Store* action box and use direct text editing to change the name of the hierarchical action box to *Check_Output*.




28. Use the  button to connect a flow between the action box and hierarchical action box.
29. Double-click on the *Check_Output* hierarchical action box to open a new child flow chart (or select the hierarchical action box and choose **Open Down** from the **File** menu).


The child flow chart is initialized in a new window tab as an action box connected by flows to a start point and end point.



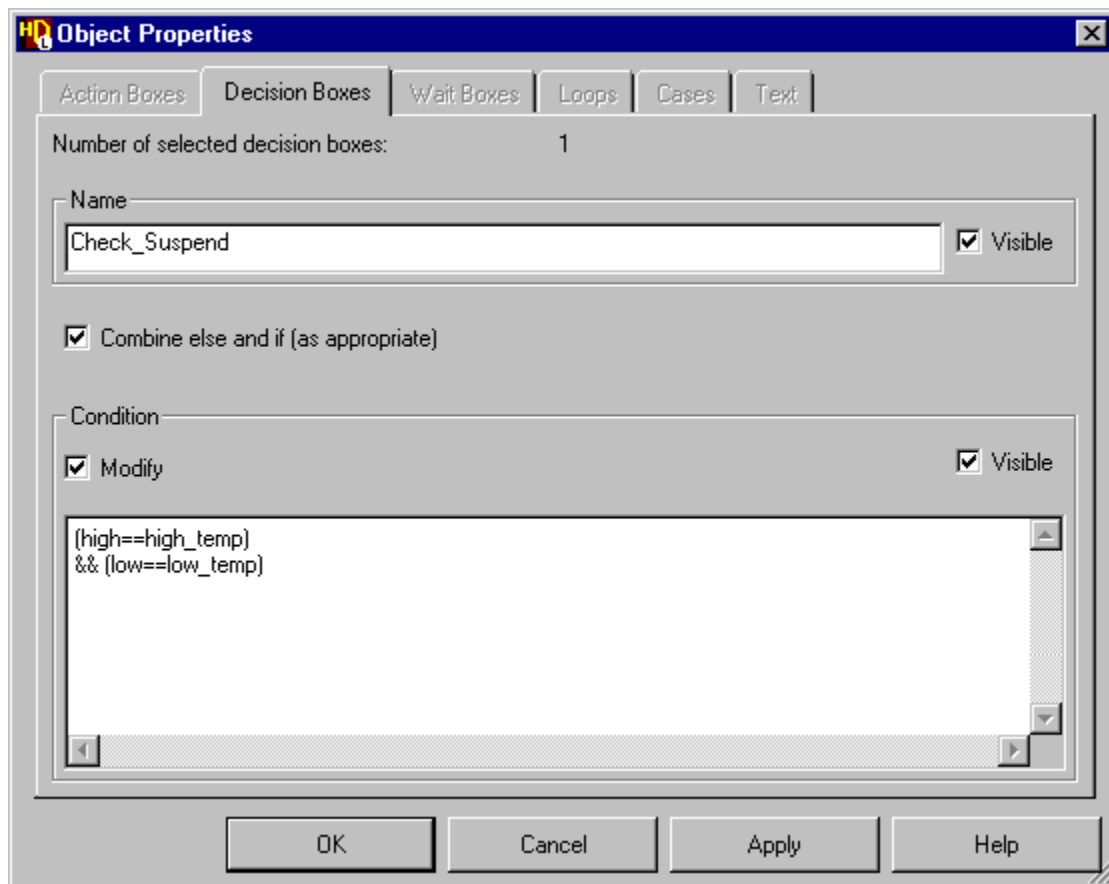
Add a Decision Box

30. Select the flows connected to the start point and end point in the child flow chart and use the **Del** key to delete them.

 You can use the **Shift** key with the left mouse button to select both flows.

31. Use the  button to add a decision box below the start point in the *Check_Output* flow chart.
32. Use the **Decision Boxes** tab of the Flow Chart Object Properties dialog box to change the name of the decision box to *Check_Suspend* and enter the following condition:


```
(high==high_temp)
&& (low==low_temp)
```



33. Select the existing action box. Use the **Action Boxes** tab of the Flow Chart Object Properties dialog box to change its name to *Pass* and enter the following actions:


Pass

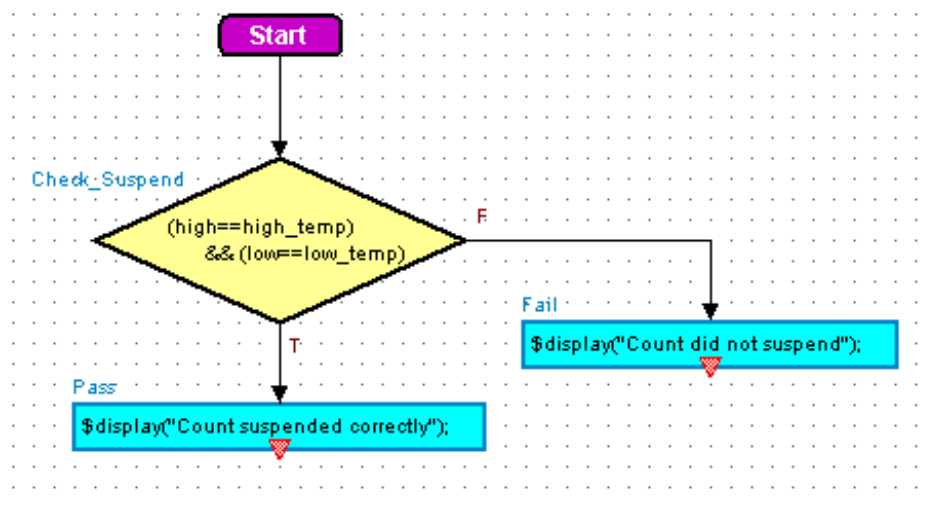
```
$display("Count suspended correctly");
```

34. Use the  button to add an action box for the False (F) condition. Use the **Action Boxes** tab to change its name to *Fail* and enter the following actions:

Fail


```
$display("Count did not suspend");
```

35. Reposition the action boxes by dragging with the mouse and use the  button to connect flows between the start point, decision box and action boxes.





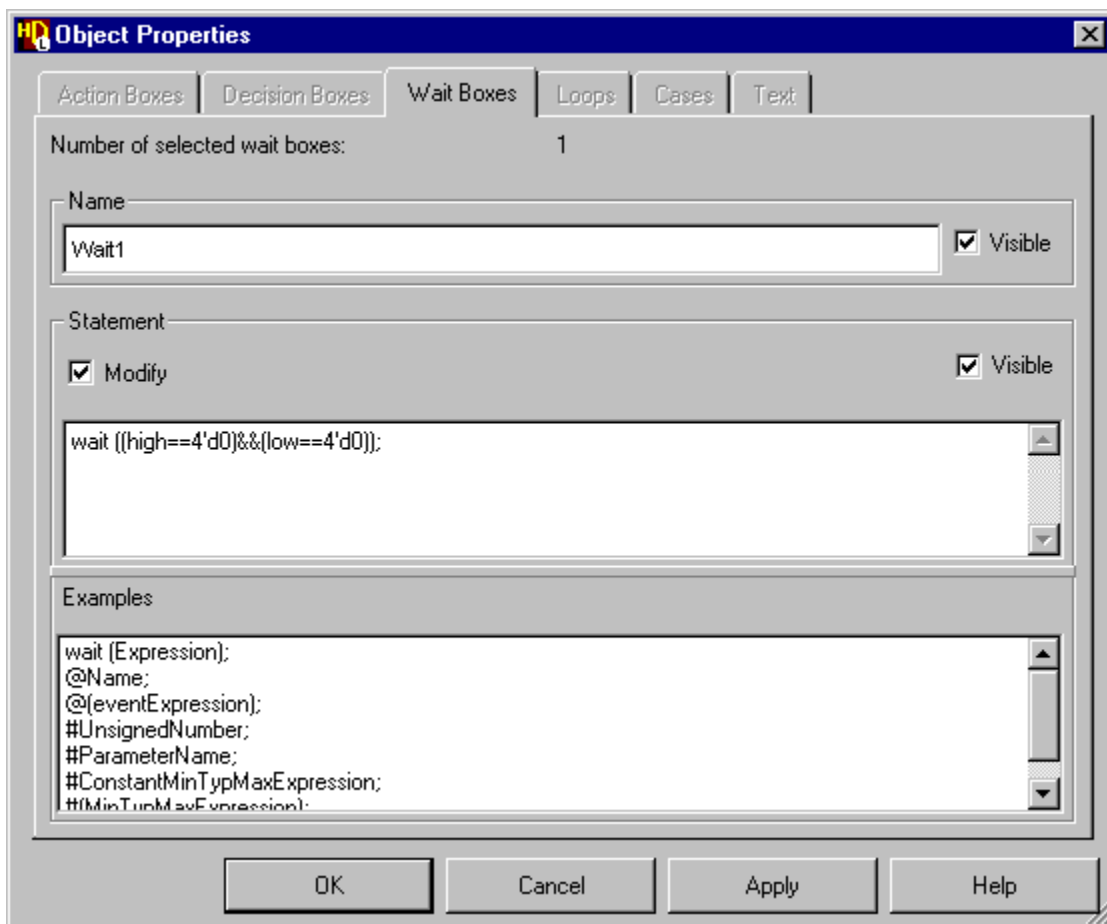
When a decision box object is selected, the popup menu includes an option to swap the True and False outputs.

Add a Wait Box

36. Use the  button to add an action box below the *Pass* action box and use the **Action Boxes** tab of the Flow Chart Object Properties dialog box to enter the following actions:

```
wait_clock(4);
stop=0;
```

37. Change the name of this action box to *Continue* and use the  button to add a wait box below it.
38. Click the mouse over the wait box (or use the  button) to display the **Wait Boxes** tab of the Flow Chart Object Properties dialog box:



39. Use the **Wait Box** tab of the Flow Chart Object Properties dialog box to rename the wait box *Wait1* and specify the following wait condition:


```
wait ((high==4'd0) && (low==4'd0));
```



You can click on one of the examples shown in the dialog box to use it as a template. Any valid expression can be used in the wait statement which must be terminated by a semi-colon.

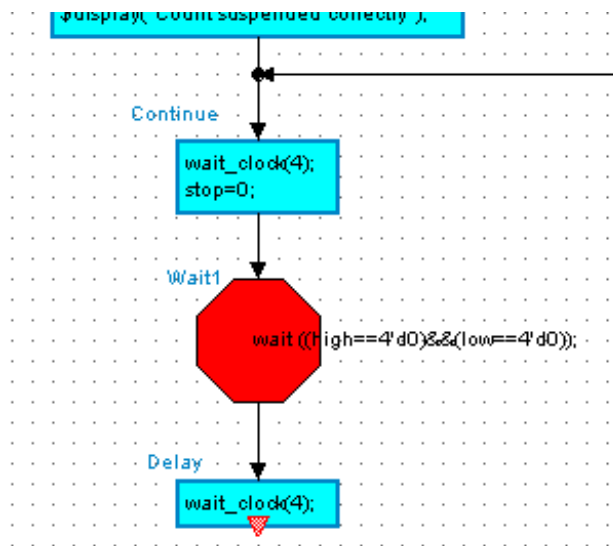
40. Use the  button to connect flows between the action boxes and wait box.

Notice that a flow join is automatically created when you connect the two flows from the *Pass* and *Fail* action boxes together.

41. Use the  button to add an action box below the wait box.




42. Use the Flow Chart Object Properties dialog box to change the name of this action box to *Delay* and enter the following action:

```
wait_clock(4);
```



Copy the Decision Tree

The child hierarchical flow chart can be completed by using similar procedures to those described in the last two topics. However, the next section can be done more quickly by copying the decision tree objects which have already been created.

43. Select the *Check_Suspend* decision box, plus the *Pass* and *Fail* action boxes and use the  button (or choose **Copy** from the popup or **Edit** menu)
44. Scroll the window down and click the  mouse button below the *Delay* action box to de-select the objects. Use the  button (or choose **Paste** from the popup or **Edit** menu) to paste a copy of the objects from the clipboard in the middle of the window. Use the mouse to drag the objects to the required position. Notice that each of the pasted objects is automatically given a unique name.
45. Use the Flow Chart Object Properties dialog box to change the names, conditions and actions where required.

```
Check_Alarm
```


```
alarm==1
```

```
Alarm_Pass
```

```
$display("Alarm asserted correctly");
```

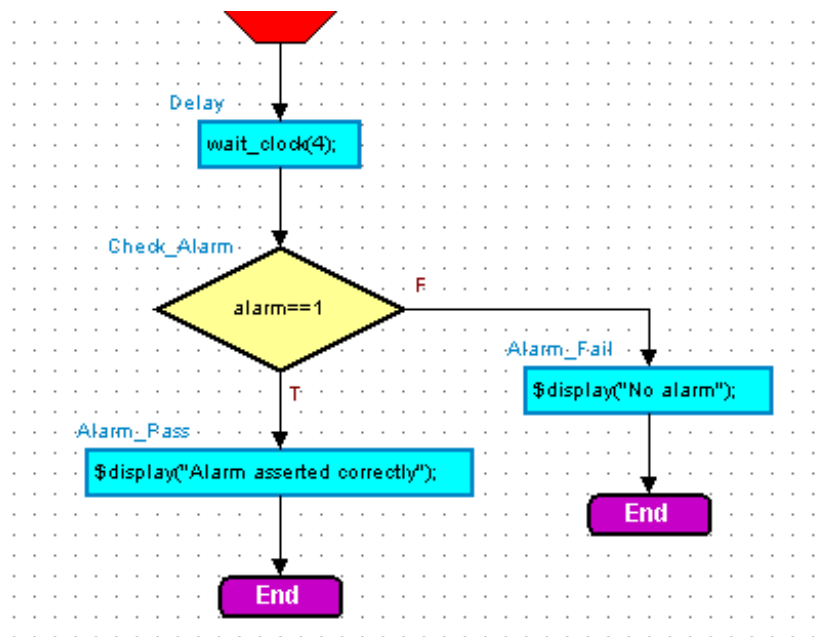
```
Alarm_Fail
```

```
$display("No alarm");
```



46. Drag the end point beneath the other objects and use the  button to connect the remaining flows on the child flow chart.



Multiple end points can be used. For example, in the following picture, a separate end point has been attached to the *Alarm_Pass* and *Alarm_Fail* action boxes.



Completing the Flow Chart

47. Use the  button (or choose **Open Up** from the **File** menu) to display the parent flow chart and the  button to add an action box below the hierarchical action box.
48. Use the Flow Chart Object Properties dialog box to change the name of this action box to *Clear* and enter the following actions:



```

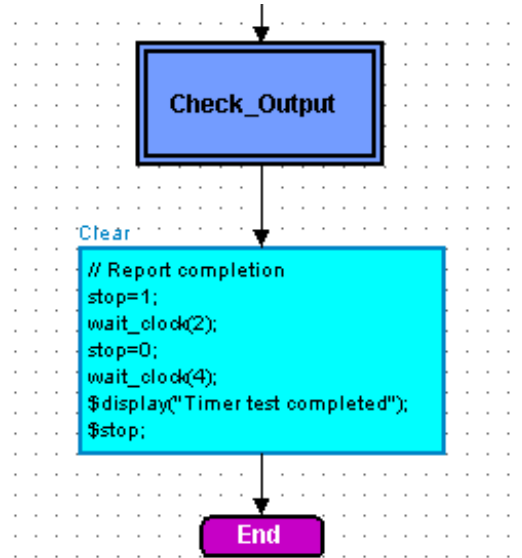
stop=1;
wait_clock(2);
stop=0;
wait_clock(4);
$display("Timer test completed");
$stop;


```



These actions are executed when the timer test has completed successfully and the \$stop system task is used to stop the simulator.

49. Use the  button to add an end point and use the  button to connect any remaining flows.



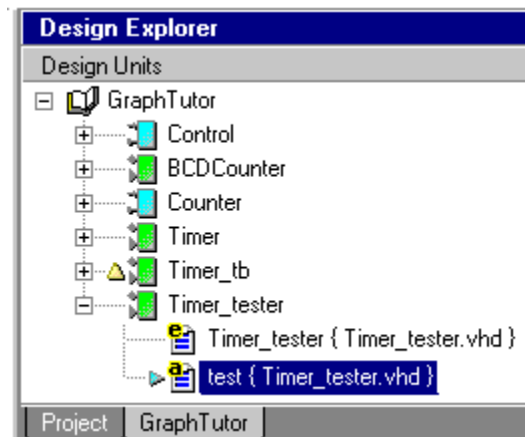
50. Complete the flow chart by editing the title, project and comments fields in the title block.
51. Use the  button to save the flow chart.


Return to [“Generate HDL for the Test Bench”](#) on page 2-71 in the main VHDL tutorial.

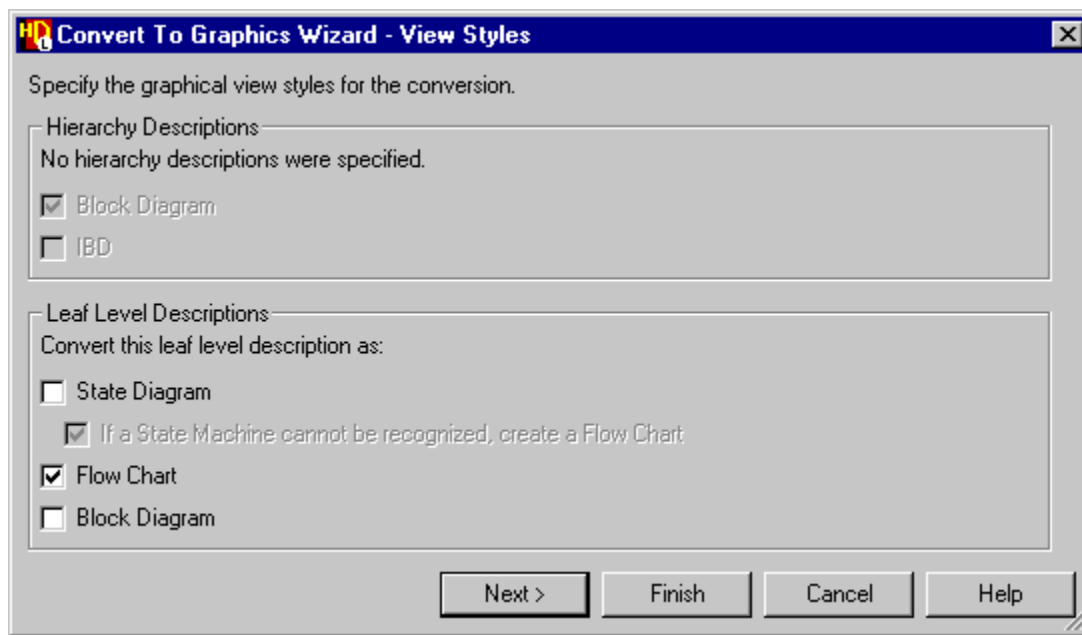
Using HDL2Graphics


If you are using **HDL Designer**, you can use HDL2Graphics to convert the HDL text view of the *Timer_tester* design unit to a graphical flow chart.

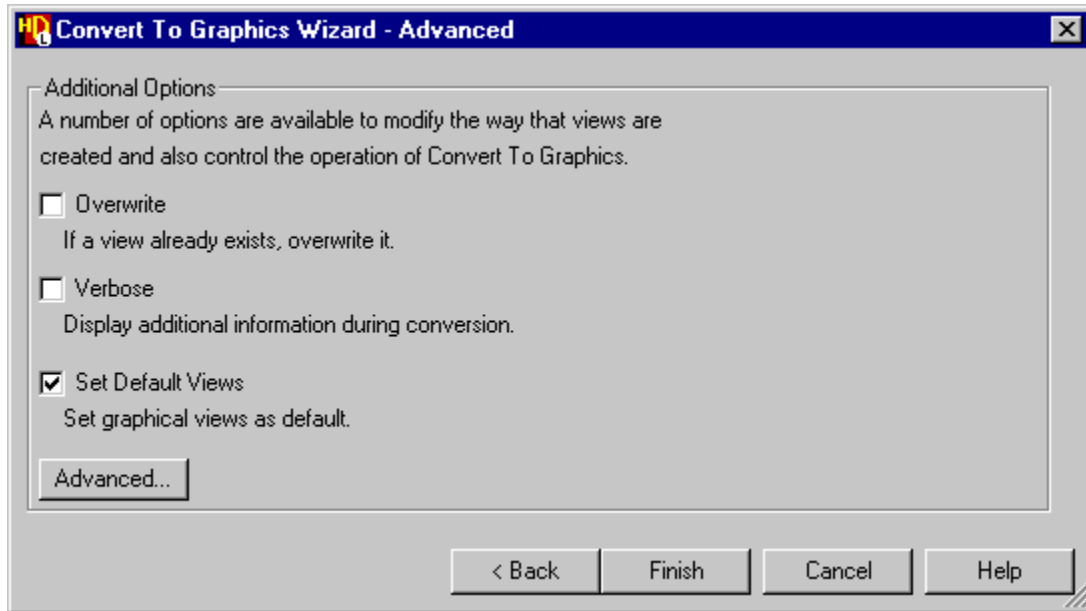
1. Select the VHDL architecture *test {Timer_tester.vhd}* for the *Timer_tester* design unit in the *Design Units* view of the design explorer:




2. Use the  button or choose **Single Level** from the **Convert To Graphics** cascade of the **HDL** menu to display the View Styles page of the Convert To Graphics wizard:



3. Set the **Flow Chart** option in the Leaf-level Descriptions section of this page and use the  button to display the Advanced page of the wizard:



4. Set the **Set Default Views** option and use the  button to convert the Verilog module view to a flow chart.

You are prompted to confirm that you wish to continue. When you confirm the prompt dialog box, the new graphical view is made the default view but the HDL text view is retained as an alternative view for the design unit.

Notice that a graphical symbol is also created for the *Timer_tester* design unit.

Return to [“Instantiate the Imported Tester” on page 2-69](#) in the main Verilog tutorial.

Trademark Information

The following names which appear in this documentation set are trademarks, registered trademarks or service marks of Mentor Graphics Corporation:

Debug Detective™, DesignBook®, Direct System Verification™, DSV™, HDL Designer Series™, HDL Author™, HDL Designer™, HDL Pilot™, HDL Detective™, HDL2Graphics™, FPGA Advantage™, Interconnect Table™, Interface-Based Design™, IBD™, Inventra™, LeonardoInsight™, LeonardoSpectrum™, Mentor™, Mentor Graphics®, ModelSim®, ModuleWare™, Precision™, Renoir™, Seamless®, Seamless CVE™, SpeedGate™ and SpeedGate DSV™.

The following names which appear in this documentation set are trademarks, registered trademarks or service marks of other companies:

Adobe, the Adobe logo, Acrobat, the Acrobat logo, Exchange, FrameMaker and PostScript are registered trademarks of Adobe Systems Incorporated.

Altera, MegaWizard and MAX+PLUS are registered trademarks of Altera Corporation; APEX and Quartus are trademarks of Altera Corporation.

ClearCase Attache is a trademark and ClearCase is a registered trademark of Rational Software Corporation.

DesignSync is a registered trademark of Synchronicity Incorporated.

FLEXlm is a trademark of Globetrotter Software, Incorporated.

Hewlett-Packard (HP), HP-UX and PA-RISC are registered trademarks of Hewlett-Packard Company.

NC-Verilog and Verilog are trademarks and registered trademarks of Cadence Design Systems Incorporated.

Netscape is a trademark of Netscape Communications Corporation.

SPARC is a registered trademark and SPARCstation is a trademark of SPARC International Incorporated.

SpyGlass is a trademark of Atrenta Inc.

Sun Microsystems and Sun Workstation are registered trademarks of Sun Microsystems Incorporated. Sun and SunOS are trademarks of Sun Microsystems Incorporated.

Synopsys, Design Analyzer, Design Compiler, FPGA Express, VCS, VCSi and VSS are trademarks of Synopsys Incorporated.

Synplify is a registered trademark of Synplicity Incorporated.

The Graphics Connection is a trademark of Square One.

Visual SourceSafe and Windows are trademarks of Microsoft Corporation.

UNIX is a registered trademark of UNIX System Laboratories, Incorporated.

Xilinx is a registered trademark and Core Generator a trademark of Xilinx, Incorporated.

Other brand or product names that appear in the documentation are trademarks or registered trademarks of their respective holders.

End-User License Agreement

**IMPORTANT - USE OF THIS SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS.
CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE SOFTWARE.**

This license is a legal "Agreement" concerning the use of Software between you, the end user, either individually or as an authorized representative of the company acquiring the license, and Mentor Graphics Corporation and Mentor Graphics (Ireland) Limited, acting directly or through their subsidiaries or authorized distributors (collectively "Mentor Graphics"). USE OF SOFTWARE INDICATES YOUR COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. If you do not agree to these terms and conditions, promptly return, or, if 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.

END-USER LICENSE AGREEMENT

- GRANT OF LICENSE.** The software programs you are installing, downloading, or have acquired with 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 you, subject to payment of appropriate license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form; (b) for your internal business purposes; and (c) on the computer hardware or at the site for which an applicable license fee is paid, or as authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or service plan purchased, apply to the following and are subject to change: (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 communicated and technically implemented through the use of authorization codes or similar devices); (c) support services provided, including eligibility to receive telephone support, updates, modifications and revisions. Current standard policies and programs are available upon request.
- ESD SOFTWARE.** If you purchased a license to use embedded software development ("ESD") Software, Mentor Graphics grants to you a nontransferable, nonexclusive license to reproduce and distribute executable files created using ESD compilers, including the ESD run-time libraries distributed with ESD C and C++ compiler Software that are linked into a composite program as an integral part of your compiled computer program, provided that you distribute these files only in conjunction with your compiled computer program. Mentor Graphics does NOT grant you any right to duplicate or incorporate copies of Mentor Graphics' real-time operating systems or other ESD Software, except those explicitly granted in this section, into your products without first signing a separate agreement with Mentor Graphics for such purpose.
- BETA CODE.** 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 you 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 your 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. If Mentor Graphics authorizes you to use the Beta Code, you agree to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. You will contact Mentor Graphics periodically during your use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of your evaluation and testing, you will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements. You agree that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceives or made during or subsequent to this Agreement, including those based partly or wholly on your 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 shall survive termination or expiration of this Agreement.

4. **RESTRICTIONS ON USE.** You 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. You 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. You shall not make Software available in any form to any person other than employees and contractors, excluding Mentor Graphics' competitors, whose job performance requires access. You shall take appropriate action to protect the confidentiality of Software and ensure that any person permitted access to Software does not disclose it or use it except as permitted by this Agreement. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, you shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive from Software any source code. You may not sublicense, assign or otherwise transfer Software, this Agreement or the rights under it, whether by operation of law or otherwise ("attempted transfer") without Mentor Graphics' prior written consent and payment of Mentor Graphics then-current applicable transfer charges. 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 licenses granted under this Agreement. The provisions of this section 4 shall survive the termination or expiration of this Agreement.

5. LIMITED WARRANTY.

5.1. Mentor Graphics warrants that during the warranty period, Software, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Software will meet your requirements or that operation of Software 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. You must notify Mentor Graphics in writing of any nonconformity within the warranty period. This warranty shall not be valid if Software has been subject to misuse, unauthorized modification or installation. MENTOR GRAPHICS' ENTIRE LIABILITY AND YOUR EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF SOFTWARE TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF SOFTWARE THAT DOES NOT MEET THIS LIMITED WARRANTY, PROVIDED YOU HAVE OTHERWISE COMPLIED WITH THIS AGREEMENT. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) SOFTWARE WHICH IS LICENSED TO YOU FOR A LIMITED TERM OR LICENSED AT NO COST; OR (C) EXPERIMENTAL BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

5.2. THE WARRANTIES SET FORTH IN THIS SECTION 5 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, WITH RESPECT TO SOFTWARE OR OTHER MATERIAL PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

6. **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 PAID BY YOU FOR THE SOFTWARE 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.

7. **LIFE ENDANGERING ACTIVITIES.** NEITHER MENTOR GRAPHICS NOR ITS LICENSORS SHALL BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF SOFTWARE IN ANY APPLICATION WHERE THE FAILURE OR INACCURACY OF THE SOFTWARE MIGHT RESULT IN DEATH OR PERSONAL INJURY.
8. **INDEMNIFICATION.** YOU AGREE 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 YOUR USE OF SOFTWARE AS DESCRIBED IN SECTION 7.
9. **INFRINGEMENT.**
 - 9.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against you alleging that Software infringes a patent or copyright or misappropriates a trade secret in the United States, Canada, Japan, or member state of the European Patent Office. Mentor Graphics will pay any costs and damages finally awarded against you that are attributable to the infringement action. You understand and agree that as conditions to Mentor Graphics' obligations under this section you must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to defend or settle the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.
 - 9.2. If an infringement claim is made, Mentor Graphics may, at its option and expense: (a) replace or modify Software so that it becomes noninfringing; (b) procure for you the right to continue using Software; or (c) require the return of Software and refund to you any license fee paid, less a reasonable allowance for use.
 - 9.3. Mentor Graphics has no liability to you if infringement is based upon: (a) the combination of Software with any product not furnished by Mentor Graphics; (b) the modification of Software other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of Software as part of an infringing process; (e) a product that you make, use or sell; (f) any Beta Code contained in Software; (g) any Software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; or (h) infringement by you that is deemed willful. In the case of (h) you shall reimburse Mentor Graphics for its attorney fees and other costs related to the action upon a final judgment.
 - 9.4. THIS SECTION 9 STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS AND YOUR SOLE AND EXCLUSIVE REMEDY WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY SOFTWARE LICENSED UNDER THIS AGREEMENT.
10. **TERM.** This Agreement remains effective until expiration or termination. This Agreement will automatically terminate if you fail to comply with any term or condition of this Agreement or if you fail to pay for the license when due and such failure to pay continues for a period of 30 days after written notice from Mentor Graphics. If Software was provided for limited term use, this Agreement will automatically expire at the end of the authorized term. Upon any termination or expiration, you agree to cease all use of Software and return it to Mentor Graphics or certify deletion and destruction of Software, including all copies, to Mentor Graphics' reasonable satisfaction.
11. **EXPORT.** Software is subject to regulation by local laws and United States government agencies, which prohibit export or diversion of certain products, information about the products, and direct products of the products to certain countries and certain persons. You agree that you will not export any Software or direct product of Software in any manner without first obtaining all necessary approval from appropriate local and United States government agencies.
12. **RESTRICTED RIGHTS NOTICE.** Software was developed entirely at private expense and is commercial computer software provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the U.S. Government or a U.S. Government subcontractor is subject to the restrictions set forth in the license agreement under which Software was obtained pursuant to DFARS 227.7202-3(a) or as set forth in subparagraphs (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19, as applicable.

Contractor/manufacture is Mentor Graphics Corporation, 8005 SW Boeckman Road, Wilsonville, Oregon 97070-7777 USA.

13. **THIRD PARTY BENEFICIARY.** For any Software under this Agreement licensed by Mentor Graphics from Microsoft or other licensors, Microsoft or the applicable licensor is a third party beneficiary of this Agreement with the right to enforce the obligations set forth in this Agreement.
14. **AUDIT RIGHTS.** With reasonable prior notice, Mentor Graphics shall have the right to audit during your normal business hours all records and accounts as may contain information regarding your compliance with the terms of this Agreement. Mentor Graphics shall keep in confidence all information gained as a result of any audit. Mentor Graphics shall only use or disclose such information as necessary to enforce its rights under this Agreement.
15. **CONTROLLING LAW AND JURISDICTION.** THIS AGREEMENT SHALL BE GOVERNED BY AND CONSTRUED UNDER THE LAWS OF OREGON, USA, IF YOU ARE LOCATED IN NORTH OR SOUTH AMERICA, AND THE LAWS OF IRELAND IF YOU ARE LOCATED OUTSIDE OF NORTH AND SOUTH AMERICA. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of Dublin, Ireland when the laws of Ireland apply, or Wilsonville, Oregon when the laws of Oregon apply. This section shall not restrict Mentor Graphics' right to bring an action against you in the jurisdiction where your place of business is located.
16. **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.
17. **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, except valid license agreements related to the subject matter of this Agreement (which are physically signed by you and an authorized agent of Mentor Graphics) either referenced in the purchase order or otherwise governing this subject matter. 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. The prevailing party in any legal action regarding the subject matter of this Agreement shall be entitled to recover, in addition to other relief, reasonable attorneys' fees and expenses.

Rev. 020826, Part Number 214231