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 Model*Sim* 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.

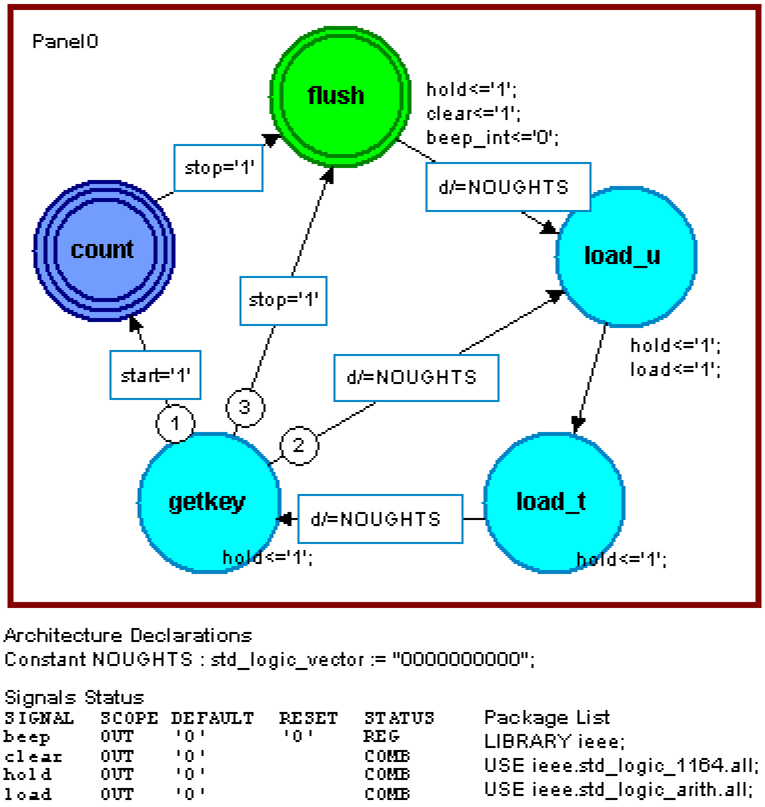
Specification

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)	

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_arith.all;
load     OUT    '0'             COMB
```

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of the *beep* signal (in the *flush* state and in the *alarm* state on the child hierarchical state diagram) have been replaced by the internal signal name *beep_int* using the default suffix *_int*.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 Help buttons for more information about each tab of the State Machine Properties dialog box.

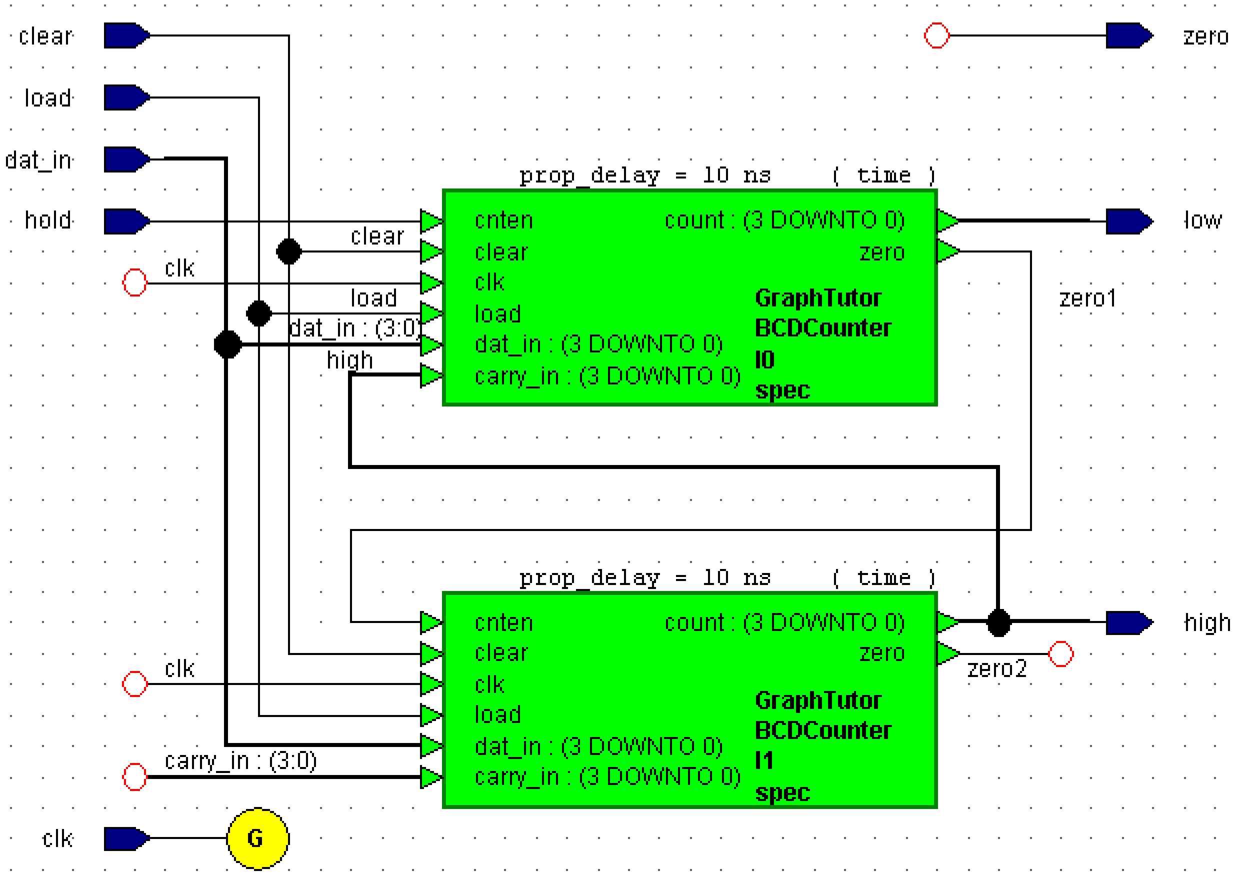
19. Select the *clear*, *load* and *dat_in* nets (or the ports with these names on instances *I0* and *I1*). Choose **Autoconnect** from the **Autoroute** cascade of the popup menu.

    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:

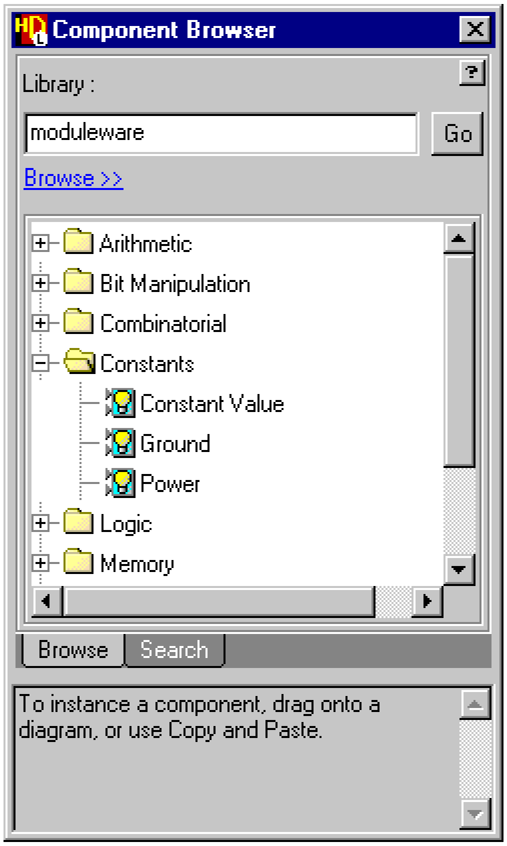


Add ModuleWare Components

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:



moduleware

To instance a component, drag onto a diagram, or use Copy and Paste.

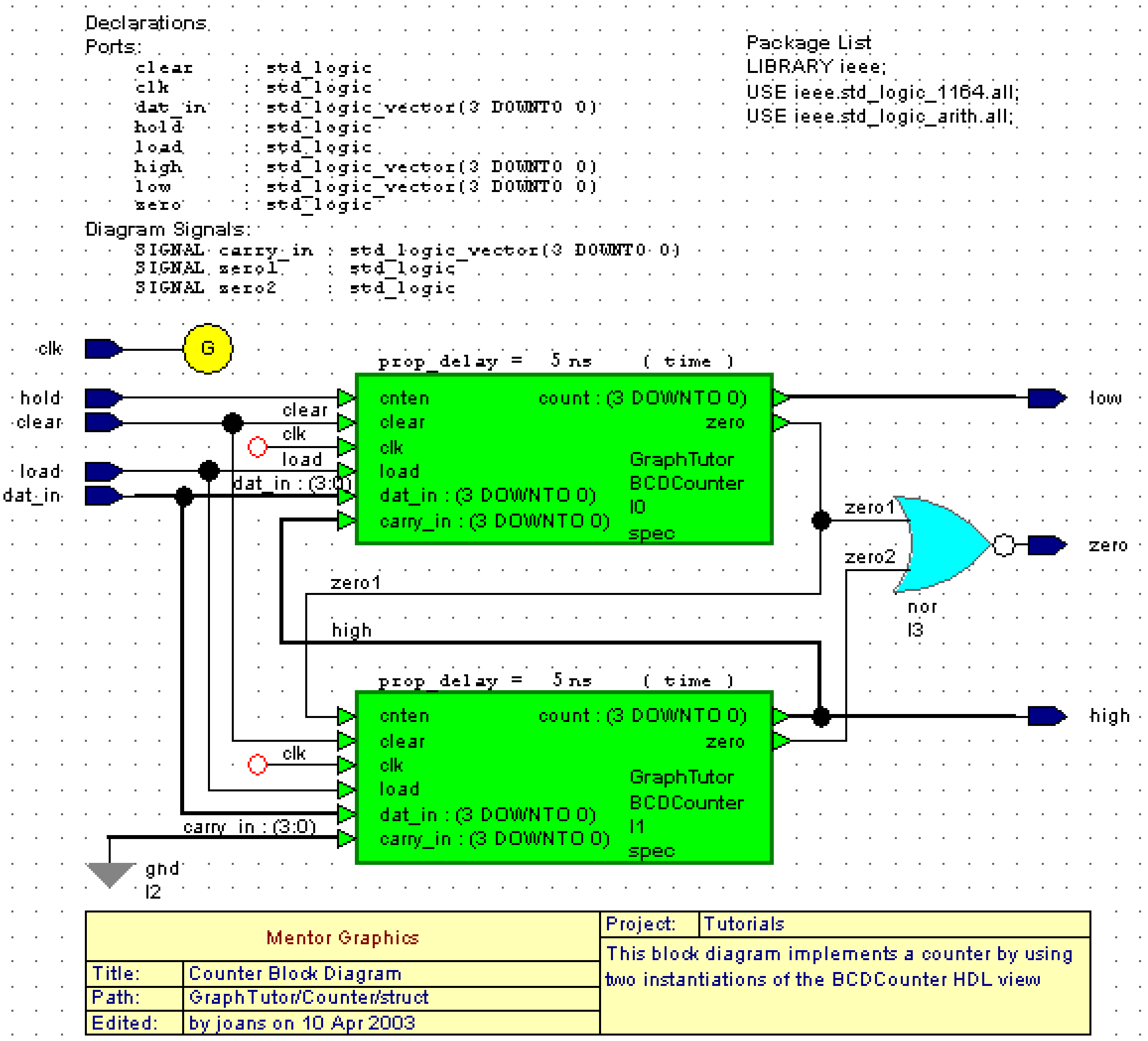
24. Use the Left mouse button to drag an instance of the *Ground* component from the Component Browser over the *Counter* block diagram and release the button to place it near the *carry_in* input to instance *I1*.

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

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

31. 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 page orientation.



```
Declarations
Ports:
    clear    : std_logic
    clk      : std_logic
    dat_in   : std_logic_vector(3 DOWNTO 0)
    hold     : std_logic
    load     : std_logic
    high     : std_logic_vector(3 DOWNTO 0)
    low      : std_logic_vector(3 DOWNTO 0)
    zero     : std_logic
Diagram Signals:
    SIGNAL carry_in  : std_logic_vector(3 DOWNTO 0)
    SIGNAL zero1     : std_logic
    SIGNAL zero2     : std_logic
```

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

Mentor Graphics		Project:	Tutorials
Title:	Counter Block Diagram	This block diagram implements a counter by using two instantiations of the BCDCounter HDL view	
Path:	GraphTutor/Counter/struct		
Edited:	by joans on 10 Apr 2003		

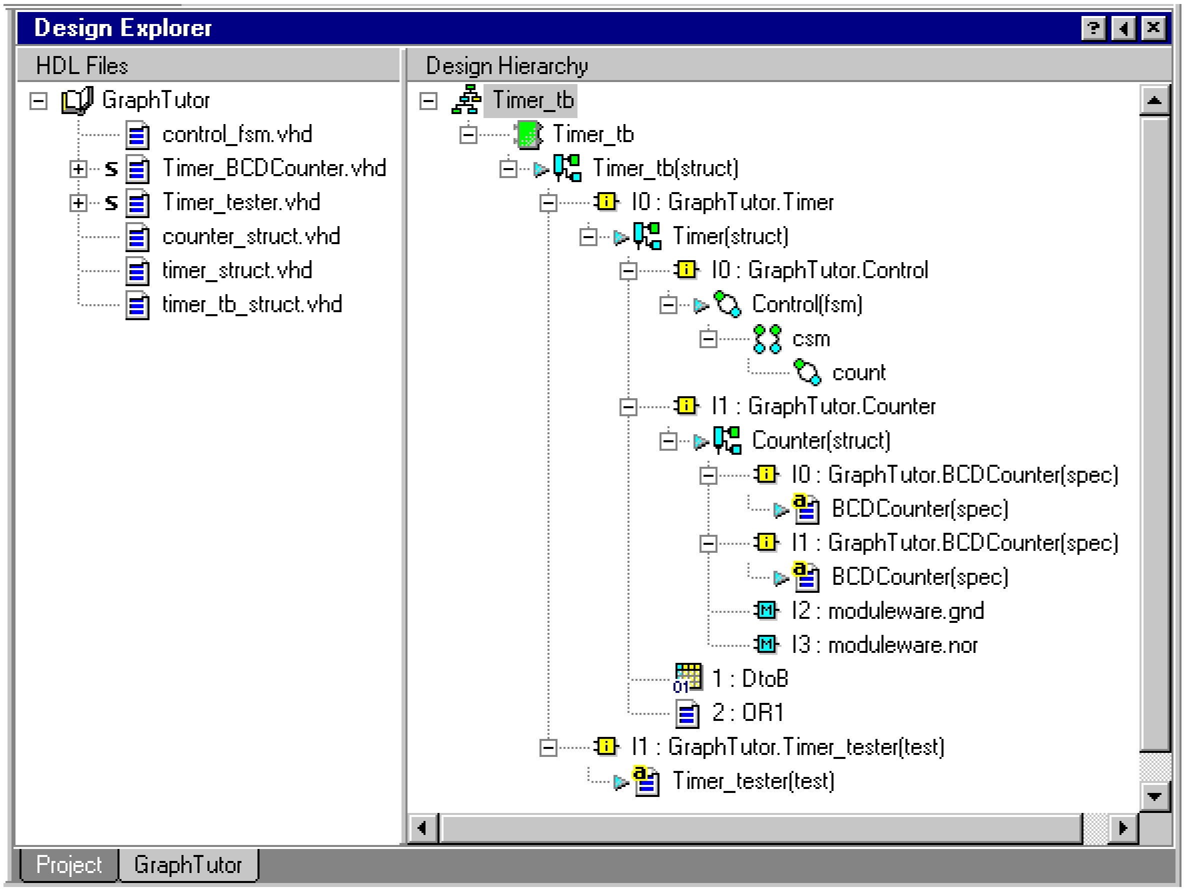
32. Use the 💾 button to save the block diagram.

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 ⊞ 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*.

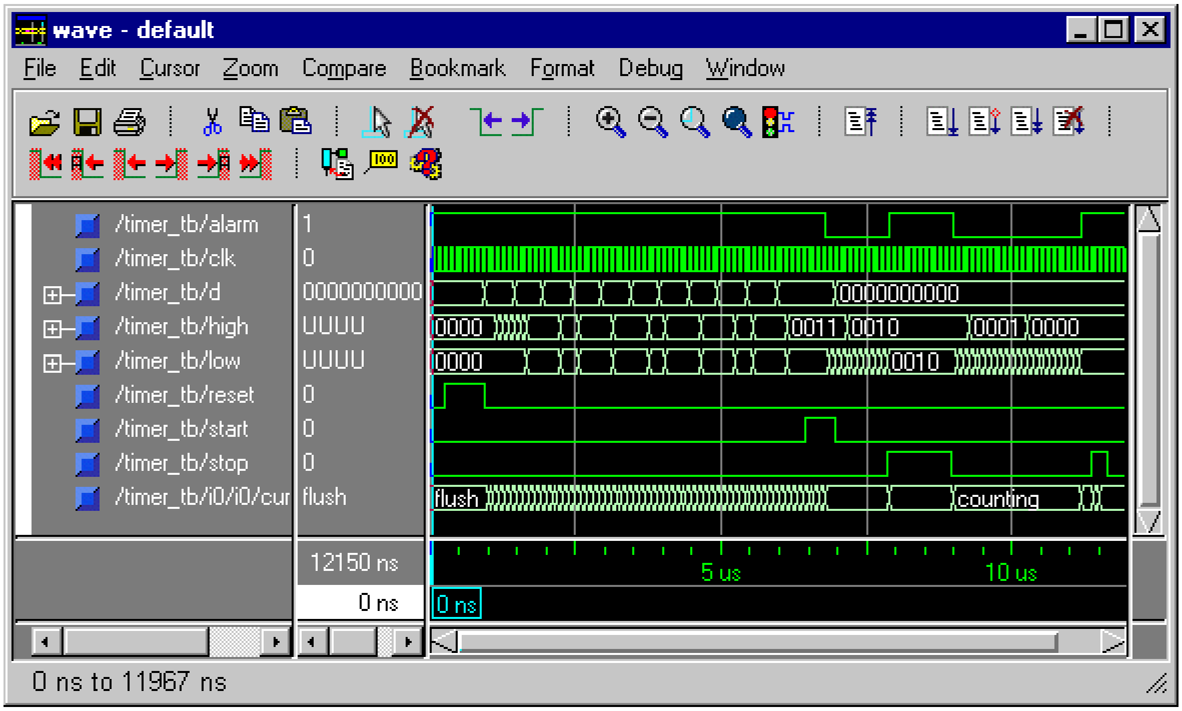


```
⊟ GraphTutor
      control_fsm.vhd
  ⊞ S Timer_BCDCounter.vhd
  ⊞ S Timer_tester.vhd
      counter_struct.vhd
      timer_struct.vhd
      timer_tb_struct.vhd
```

```
⊟ Timer_tb
  ⊟ Timer_tb
    ⊟ Timer_tb(struct)
      ⊟ I0 : GraphTutor.Timer
        ⊟ Timer(struct)
          ⊟ I0 : GraphTutor.Control
            ⊟ Control(fsm)
              ⊟ csm
                  count
          ⊟ I1 : GraphTutor.Counter
            ⊟ Counter(struct)
              ⊟ I0 : GraphTutor.BCDCounter(spec)
                  BCDCounter(spec)
              ⊟ I1 : GraphTutor.BCDCounter(spec)
                  BCDCounter(spec)
                I2 : moduleware.gnd
                I3 : moduleware.nor
            1 : DtoB
            2 : OR1
      ⊟ I1 : GraphTutor.Timer_tester(test)
          Timer_tester(test)
```

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.

.



/timer_tb/alarm 1
/timer_tb/clk 0
/timer_tb/d 0000000000
/timer_tb/high UUUU
/timer_tb/low UUUU
/timer_tb/reset 0
/timer_tb/start 0
/timer_tb/stop 0
/timer_tb/i0/i0/cur flush

12150 ns
0 ns

0 ns to 11967 ns

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).