

## **ACTEL CORPORATION**

## **ACTEL TRAINING**

## VHDL LAB GUIDE FOR LIBERO IDE ver2.3



# Introduction to the Actel VHDL Design Flow

## Introduction

This guide will take you through the design flow for VHDL using Actel Libero IDE version 2.3. It explains briefly how to use the software tools and provides information about the example design.



#### Libero<sup>™</sup> IDE HDL Design Flow

Actel VHDL Design Flow

## **Overview**

### Purpose

The purpose of this lab is to become familiar with the Actel VHDL design flow. For this exercise, we will implement a 16-bit loadable counter with an asynchronous reset and synchronous enable in an AX500 FPGA.

### Tools

For this lab, you will use the following tools:

- Libero IDE ver 2.3
- WaveFormer Lite 8.9
- ModelSim for Actel ver 5.6b
- Synplicity ver 7.2
- Designer R1-2003

### Function

16 bit synchronous counter triggered with the positive edge of the clock

### Pin list

ENABLE = enable count active high

RESETn = asynchronous reset of the counter active low

CLK = master clock

LOAD = parallel load of the counter active high

DATA = 16 bit data input to counter

COUNT = 16 bit counter output



Counter block diagram



# **Creating a project in Libero IDE**

## Creating a Libero IDE Project and entering the source file



Start Libero IDE by double clicking on the Actel Libero IDE shortcut on your desktop or by clicking Start > Programs > Libero IDE 2.3 > Libero IDE Design Environment.

From the File menu click New Project. The New Project dialog box appears, as shown.

Enter the following in the New Project dialog box:

**Project Name:** Enter counter16 in the Project Name field (note that a folder named counter16 will be created under the specified project location)

Location: Browse to C:\Actelprj

**Family:** Select Axcelerator from the Family dropdown list box

HDL: Select VHDL

Click *OK*. The project "counter16" is created and opened in Libero IDE.

New Project	×
Project <u>N</u> ame:	
counter16	
Project <u>L</u> ocation:	
C:\Actelprj\counter16 Brows	e
Eamily: HDL Type:	
Axcelerator  Verilog	
OK Cancel Help	

#### Creating the VHDL source file

From the file menu, click **New**. In the New dialog box, select VHDL Entity and enter *counter16* as the name. Click **OK** to open the HDL editor.

In the HDL editor, enter the description of a 16 bit counter as described in the overview section.

Note: you can copy the description of the counter from Appendix A of this document and paste it into the Libero IDE HDL editor.

New	×
File <u>T</u> ype:	ОК
Schematic ACT gen macro	Cancel
VHDL Entity VHDL Package File Stimulus Stimulus HDL File	Help
<u>N</u> ame:	
counter16	

Save the counter description by selecting **Save** from the Libero IDE **File** menu. The counter description will be visible in the HDL editor and on the Design Hierarchy tab as shown in the figure below:



Select the File Manager tab in the Libero IDE Design Explorer window. Select *counter16* under HDL Files, then right mouse click and select **Check HDL file**. If your counter description has no syntax errors,

"Syntax checking for C:\Actelprj\counter16\hdl\counter16.vhd is successful" will appear on the **Output** tab in the Libero IDE GUI.

Correct any syntax errors that exist in your counter description and save the file.



4



## **Performing functional simulation**

In this section you will generate stimulus for the counter and perform a functional (pre-synthesis simulation).

### **Creating Stimulus with WaveFormer Lite**

WaveFormer Lite generates VHDL testbenches from drawn waveforms. There are three basic steps for creating test benches using WaveFormer Lite and Actel Libero IDE:

- Import Signal Information
- Draw WaveForms
- Export the VHDL Testbench

To launch WaveFormer Lite and import signal information into it, go to the **Design Hierarchy** tab in the Libero IDE Design Explorer Window and highlight *counter16*. Right click and select **"Create Stimulus"** to invoke WaveFormer Lite.



Note: if counter16 does not appear in bold font on the Design Hierarchy tab, highlight counter16, right mouse click and select "**Set As Root**".

WaveFormer Lite launches, with the port signals appearing in the Diagram window.

For additional information on using WaveFormer Lite, refer to the WaveFormer Lite User's Guide (wflite.pdf) which is contained in the docs folder in your Libero IDE installation path.

### **Generating Stimulus for the Counter**

In this step, you will generate stimulus and a testbench for the *counter16* design using WaveFormer Lite.

Create the clock signal by clicking the *CLK* signal in the diagram window then right mouse clicking and selecting "**Signal(s) <-> Clock(s)**".

A Clock waveform will appear in the diagram window. Double click on the clock signal name in the diagram window to open the Signal Properties window. Click the Clock Properties button to open the "Edit Clock Parameters" window. Specify the clock parameters to generate clock signals.

Using the instructions above, create a clock signal with the following properties:

- CLK 100 MHz
   Duty Cycle: 50% (default)
- Starting Offset: 0 (default)
   Invert (starts low) unchecked

Accept defaults for all other clock parameters. Click OK to close the Clock Properties dialog box.

Click **Apply** then **OK** in the Signal Properties dialog box to create the clock waveform.

Draw waveforms for the other signals as listed below:

•	RESETn - low 0 ns - 35 ns	• Load - low 0 ns - 775 ns
	high 35 ns - 1.2 us	high 775 ns - 805 ns
		low 775 ns - 1.2 us
•	Enable - low 0 ns - 65 ns	• Data - 0000 0 ns - 765 ns
	high 65 ns - 1.2 us	FFF8 765 ns - 1.2 us

Your waveforms should appear as shown in the figure on the next page.

ACTEL TRAINING PROG	RAM
---------------------	-----

🕂 Diagram - untitled	1.btim*
Add Signal Add Bus Add Clock Add Space	n Hold Text Marker HJGH LOW TRI VÁL INVal WHI WLO HEX Q- Q R
944.0ns 327.0ns	0ns  100ns  200ns  300ns  400ns  500ns  6 <mark>0</mark> 0ns  700ns  800ns  900 <mark>n</mark> s  1.0us  1.1us  1.2
CLK	
RESETn	
LOAD	<u></u>
ENABLE	
DATA[15:0]	0000 <u>(</u> FFF8
COUNT[15:0]	
Image: A transformed and tr	

WaveFormer Lite Diagram Window

Save the timing diagram by clicking **Save** from the WaveFormer Lite **File** menu.

In the Save As dialog box, enter *counter16.btim* as the file name then click **Save** to save the timing diagram.

Generate the VHDL testbench by selecting **Export Timing Diagrams As** (Export menu). In the Save As dialog box, select **VHDL Wait with Top Level TestBench (\*.vhd)** in the "Save As Type" pull-down menu. Enter *counter16\_tb.vhd* in the filename box and click **Save** to generate the testbench. After WaveFormer Lite creates the file, it will display its contents in the Report window so that you can quickly verify that the file is correct.

Exit WaveFormer Lite by selecting **Exit** (File menu). Select **Yes** when prompted about closing all text files.

The waveform file and the testbench will appear on the File Manager tab in the Libero IDE Design Explorer window.

Save As			?
Savejn:	🔄 stimulus	<u> </u>	
File <u>n</u> ame:	counter16.btim		Save
	Timing Diagram - Binary (*.btim)	-	
Save as type.	Timing Diagram - binary (".bum)	<u> </u>	Cancel
Gave As			?
G <b>ave As</b> Savejn:	🔄 stimulus	• <b>E</b>	
r	🔁 stimulus	▼ £	
r	🔄 stimulus	▼ Ē	
r	itimulus	V È	
r	🔄 stimulus	• È	
r	🔄 stimulus	¥ È	
r	atimulus	T E	
r	Counter16_tb.vhd		

Erroject Design Files
Block Symbol Files
Schematic Files
🛨 🚥 HDL Files
Vhdl Package Files
ACTgen Macros
Implementation Files
⊡····· Stimulus Files
counter16.btim
🛄 counter16_tb.vhd
Design Hierarchy File Manager

## Performing pre-synthesis simulation with ModelSim for Actel

To perform pre-synthesis simulation, double click the **ModelSim Simulation** button in the Libero IDE Process window, or right mouse click on *counter16* (*Design Hierarchy* tab) in the Design Explorer Window and select **Run Pre-Synthesis Simulation**.

A dialog box will open indicating that no testbench stimulus is associated with counter16. Select the **Associate stimulus** radio button and click **OK**.

Warning	×
No test bench stimulus is associated with counter16. What would you like to do?	
Associate stimulus	
Start ModelSim without loading stimulus	
OK Cancel Help	

In the Select Stimulus dialog box, highlight counter16\_tb.vhd and click **Add** to add the testbench to the Associated Files box.

Click **OK** to close the Select Stimulus dialog box and launch the ModelSim for Actel simulator.

Select Stimulus Dialog			×
Associate stimulus files for counter16 v Use the Up and Down buttons to spec		order of the simulator.	
Stimulus Files in the project:		Associated Files:	<b>† f</b>
counter16 tb.vhd			
	Add 🔶		
	+ Remove		
	ОК	Cancel	Help
			пер

The ModelSim for Actel VHDL Simulator will open and compile the source file and the testbench.

ModelSim ACTEL 5.6b - Custom Actel Version	
<u>File Edit View Compile Simulate Tools Window H</u>	elp
🗇 😅   🖻 🛍   If 🔢 100 🕂 Il II II I	₩ <sup>4</sup> 0 0 <sup>4</sup> 1
testbench: testbench(tbgeneratedcode) stimulus_0: stimulus(generatedcode) counter16_0: counter16(rtl) Package std_logic_unsigned Package std_logic_arith Library_sim	<pre># Loading entity stimulus # Loading package std_logic_unsigned # Loading entity counter16 # vsim -t 1ns presynth.testbench # Loading d:/Libero/Model/win32acoem//std.standar d # Loading d:/Libero/Model/win32acoem//std.textio(b ody) # Loading d:/Libero/Model/win32acoem//ieee.std_lo gic_1164(body) # Loading d:/Libero/Model/win32acoem//ieee.std_lo </pre>
Now: 1 us Delta: 4 sim:/testb	ench _//.

When the compilation completes, the simulator will run for 1 us and a Wave window will open to display the simulation results.



ModelSim for Actel Wave Window

Scroll in the wave window to verify the counter works correctly. Use the zoom buttons to zoom in and out as necessary. The radix for the data and count signals can be changed to improve readability.



Wave window Toolbar Buttons.

Exit the simulator by selecting **Quit** from the **File** menu in the ModelSim for ACTEL 5.6b window. Enter **Yes** in the **Quit VSIM** dialog box.



# Synthesizing the counter

## **Creating a Synplicity project**

In this section, you will synthesize the counter design with Synplicity to create an EDIF netlist.

Invoke Synplify by double clicking the **Synplify Synthesis** button <sup>••</sup> in the Libero IDE Process Window or by right mouse clicking on *counter16* in the Libero IDE Design Explorer Window and selecting **Synthesize**. The Synplicity main window will open.

😘 Synplify - [C:\Actelprj\counter16\hdl\counter16_syn.prj]			_ 🗆 ×
Eile Edit View Project Bun HDL Analyst Options Window Help			<u>-8×</u>
🛛 P 🖹 🗆 🗲 🖬 🎒 🎒 👗 🔍 🛱 🖄 🖻	:  ∧ ∥⊕ :	> 🗿 🔁 ∻ 📗 🖬	turi 🖉
Synplify® Source Files		Syn <i>plicity</i>	
Add C:\Actelpri\counter16\hdl	hdl	Simply Better Results	
Change counter16_syn (project) Change whdl Edit		Frequency (MHz) 100 Symbolic FSM Compiler Resource Sharing	۲ ۲
Result File			- 1
Change counter16.edn			
Target Change Actel Axcelerator : AX125 : Std, maxfan: 10			- 1
Run View Log Rea	ady		
C:\Actelpri\c			

Synplicity main window

Change the target device by clicking the **Change** button.



Fanout Guide: 10 (default)

Hard Limit to Fanout: off (unchecked)

Disable I/O Insertion: off (default).

The "Options for Implementation: counter16\_syn: hdl" dialog box will open. In the device tab, confirm the following are set and click **OK**:

•

٠

•

- Technology: Actel Axcelerator
- Part: AX500
- Speed Grade: -3

In Synplicity's main window:

- Set the Frequency to 100 MHz
- Turn resource sharing off (unchecked)
- Symbolic FSM Compiler turned off (unchecked)

Click the **RUN** button. Synplify will now compile and synthesize the *counter16* design into a file called *counter16.edn*. When the **Ready...** on the main user interface in Synplify changes to **Done...** the design has been successfully mapped to the Axcelerator family.

The resultant EDIF file, *counter16.edn*, and a structural VHDL netlist will be visible under Implementation Files on the Libero IDE File Manager tab in the Design Explorer Window.

Frequency (MHz)	100	•
Symbolic FSM Cor		
Resource Sharing		

Done!
🖃 🗝 Project Design Files
Block Symbol Files
Schematic Files
🕀 🚥 HDL Files
······ Vhdl Package Files
ACT gen Macros
📮 🚥 Implementation Files
🔤 🔂 counter16.edn
counter16.vhd
🗄 🗝 Stimulus Files
Design Hierarchy File Manager

.

Click on the **View Log** button and scroll through the log file to answer the following questions:

#### Utilization

Combinational Cells: \_\_\_\_\_ Sequential Cells: \_\_\_\_\_

#### Frequency

Estimate Frequency:\_\_\_\_\_ MHz

\_\_\_\_\_ns



## Simulating the counter structural VHDL netlist

In this section, you will simulate the structural VHDL netlist of the counter using the VHDL testbench that was created in section 2.

Click the **ModelSim Simulation** button in the Libero IDE Process window, or right mouse click on *counter* (*Design Hierarchy* tab) in the Design Explorer Window and select **Run Post-Synthesis Simulation**.

The ModelSim for Actel VHDL Simulator will open and compile the source file and the testbench.



When the compilation completes, the simulator will run for 1 us and a Wave window will open to display the simulation results.

Scroll in the wave window to verify the counter works correctly. Use the zoom buttons to zoom in and out as necessary. The radix for the data and count signals can be changed to improve readability.

🔫 wave - default											_	□×
<u>File Edit View Insert Form</u>	nat <u>T</u> ools <u>W</u> indow											
😅 🖬 🎒   👗 🖻 🛍 (	🛤   👃 🔏 🕒 :		Ēg.   🍳 (	ର୍ 🔍 👫		↓ ⊒¢ ⊒¥	<b>34</b>   3 <del>4</del>					
📕 /testbench/clk	1	າດແບບບານ				www.www						$\square$
📕 /testbench/resetn 📕 /testbench/load	1 0									η		
/testbench/enable	1 11111111111111000	000000000	000000						<u>)</u> (111	11111111111	000	
	0000000000001011										000000000	
	1000 ns		2	1 00	4(	)0	61	1 00	81 81	1 )0	11u	us I
	0 ns	0 ns										
	• •	$\triangleleft$										$\geq$
0 ns to 1025 ns												11



# **Place and Routing the COUNTER**

The next step in the design flow is to use Actel's Designer to implement the counter design in an AX500.

## **Opening Designer R1-2003**

Double click the **Designer Place-and-Route** button in the Libero IDE Process window, or right mouse click on *counter16* (*Design Hierarchy* tab) in the Design Explorer Window and select **Run Designer**. This starts the Designer place & route tool.



Click the **Compile** button to begin the process. The **Device Selection Wizard** will open. Select the AX500 and the 484FBGA package. Accept the default speed grade and die voltage and click **Next**.

Next, the **Device Selection Wizard – Variations** window opens. Accept the default settings and click **Next**.

The **Device Selection Wizard – Operating Conditions** window opens. Accept the default settings and click **Finish**. Designer will then read the netlist, checking for errors and compile it for the next step. Once completed, the **Compile** button will turn green.





Designer GUI after compiling the counter16 design



## **Assigning Pin locations**

Next use the optional step and define the pin-out of the device *prior* to place and route. The Pin Editor can be used to make and edit I/O macro pin assignments and select I/O pin standards for families which support multiple I/O standards.

Click the **PinEdit** button to open the **Pin Editor**.

File	counter16 Edit Viev										<u>- 0 ×</u>
			1 - 1 - 1 - 1	20							
								Unassigne CLK COUNT(0) COUNT(1) COUNT(2) COUNT(3) COUNT(4) COUNT(5) COUNT(5) COUNT(6) COUNT(7) COUNT(8) COUNT(8) COUNT(9)		Ass	igned
<b>▲</b> Seler	 ction Mode						<b>•</b>				•
×	Port	Name	Macro Cell	Pin #	Fixed	Bank Name	I/O Standard	Output Drive (mA)	Slew	Resistor Pul	l Input Del 🔺
Ĥ	1 LC	)AD	ADLIB:INBUF	Unassigned	Г		LVTTL			None	
	2 COL	NT(0)	ADLIB: OUTBUF	Unassigned	Г		LVTTL	24	High	None	
		FA(9)	ADLIB:INBUF	Unassigned	Г		LVTTL			None	
		ABLE	ADLIB:INBUF	Unassigned	Г		LVTTL			None	
	5 COU	VT(10)	ADLIB:OUTBUF	Unassigned			LVTTL	24	High	None	
	•				_						
								FAM: axceler	ator DI	E: AX500 PKG	: 484 FBGA 🏼 🎵

In the **PinEdit** window, the following color-coding scheme is used to denote pin type:

- Package pins shown in green with a black center are available for signals.
- Package pins shown in red are reserved for power and ground.
- Package pins shown in blue with a black center are for reserved pins (e.g. JTAG and Probe pins) and for special pins such as clock inputs.

Selecting a signal pin will cause the pin outline to change from green to white. After assigning signals to a pin, the pin outline will change to yellow indicating a fixed pin assignment.

The Axcelerator Family supports multiple I/O standards and I/Os are grouped onto I/O banks. The I/O banks are color coded for quick identification. Colors can be customized using the Pin Editor Color Manager if desired.

Drag the Signal CLK from the **Unassigned** pane and drop it on the package pin labeled HCLKAP (pin E10). Note that CLK now moves to the **Assigned** pane and the spreadsheet below is updated to reflect the assignment of the CLK signal.



Drag and drop the RESETn signal to the package pin labeled CLKEP (pin V13). Note that each signal moves to the **Assigned** pane and the spreadsheet is updated as it is placed.

Assign other signals to unused green pins. When you are finished placing all the pins, click **File > Close**, to quit **PinEdit** and return to Designer. Answer **Yes** when prompted if you want to save changes you made in Pin Edit.

## Layout



Now you will place and route the counter design – 100% automatically. To invoke the place & route tool, click the Layout button. Accept the default settings in the **Layout Options** window and click **OK**. If you had added timing constraints, the Timing-Driven option would be available. For designs which were previously place & routed that have minor changes, you could select **On** or **Fix** under the **Incremental** options in the Layout window.

Layout will place and route our design, and the Layout button will turn green when completed.

### Timer



Next, we will do a quick timing analysis on the counter design. Invoke Timer by clicking the **Timer** icon. The Timer window will open showing a speedometer with the max clock frequency for the counter design. Note the frequency, temp and speed grade. When finished, **File > Close**, to quit.

What frequency did Timer indicate the counter would run?

Counter16 - Timer
Select Clock => CLK
Summary Clocks Paths Breaks
CLK Frequency
172 230
57 Frequency
<b> 203</b> MHz
Actual: 203.42 MHz
Required: MHz Expand
Maximum Delay in the CLK domain between all
Actual(ns) Required (ns)
Input Ports to Registers: 3.87
Registers to Output Ports: 3.45
Input Ports to Output Ports:
Set
Ready [Temp: 70] Volt: 1.425 Speed: -3 //

## **Exporting a Timing Report**

You can export a timing report for the counter design from Designer. From Designer's main menu, select **Tools > Reports**. In the Report Types dialog box, select **Timer** in the drop down menu then click **OK**. In the Timing Report Dialog box, click **Options**.

Select sort by Actual. In the Longest/Shortest Path(s) field, enter 10. Click OK.

Click **OK** in the Timing Report dialog box to accept the other default settings.

A Timing Report Dialog window will open as shown on the next page.

In the Timing Report window, select **File > Save As**. Name the file **counter16**.*rpt*.

Select **File > Close** to close the Timing Report window.

🔁 counter16 - Timer Report	
<u>F</u> ile <u>H</u> elp	
Design: counter16	
Family: axcelerator	
Die: AX500	
Package: 484 FBGA	
Radiation Exposure: O KRad	
Temperature: COM	
Voltage: COM	
Speed Grade: -3	
Design State: Post-Layout	
Timing: Worst Case	
Path Tracing: Longest Paths	
Break at Clk/G pins: True	
Break at Preset/Clr pins: True	
Break at Data pins of Latchs: True	
Section Clock Frequency Actual Required ClockName	
Actual Required ClockName 203.42MHz N/A CLK	
End Section	
End Section	
Section \$Inputs() to \$Outputs()	
No Paths found	
End Section	
Section \$Inputs() to \$Registers(CLK):\$DataPins()	
Delay(ns) Slack(ns) Pins	
3.48 N/A From: LOAD	
To: COUNTZOZ 3/UO:D	
3.44 N/A From: LOAD	-

Timing report for the counter

### **Back-annotate**

To do timing simulation using post-layout results, you need to generate the necessary files: postplacement netlist and an SDF (Standard Delay Format) file with actual timing numbers from our place & route. Click the **Back-Annotate** button and the Back-Annotate window will open.

Accept all defaults and click **OK** (see figure to right).



### Saving your design files

_	

Back

Annotate

Be sure the save your work! When finished, click **File** > **Save** then **File** > **Exit**, to close Designer.

After exiting Designer, the Actel database (*counter16.adb*) and the timing information (*counter16\_ba.sdf*) will be visible in the Libero IDE Design Explorer window on the File manager tab under Implementation Files.

🖃 Project Design F	iles
Block Symbo	ol Files
Schematic F	ïles
🗄 🗝 HDL Files	
Whdl Packag	ge Files
ACT gen Ma	cros
🕂 🛱 🛄 İmplementati	ion Files
🛛 🔂 cour	nter16.edn
counter	16.vhd
🛛 🔂 cour	nter16.adb
counter	16_ba.vhd
counter	_
Stimulus File	-
L	
Design Hierarchy	File Manager



# **Back-Annotated Timing Simulation**

## **Invoking the Simulator**

In this section, you will simulate the structural VHDL netlist of the counter using the VHDL testbench that was created in section 2 and the actual timing numbers (SDF) exported from Designer in section 5.

Click the **ModelSim Simulation** button in the Libero IDE Process window, or right mouse click on *counter16* (*Design Hierarchy* tab) in the Design Explorer Window and select **Run Post-Layout Simulation**.

The ModelSim for Actel VHDL Simulator will open and compile the source file and the testbench.



Observe the waveforms in the Wave window and confirm that the counter operates correctly and your results match the results from Sections 2 and 4. Change the radix of the signals and use the zoom controls as necessary to match the results shown below.

🔫 wave - default												_ 🗆 ×
<u>File E</u> dit <u>V</u> iew <u>I</u> nsert F <u>o</u> rn												
📂 🔒 🎒   👗 🖻 🛍	🛤   占 🔏 ገ	± 🛃   💽	୮ଘ   🔍 ପ୍	🔍 👫 📗	IT I I	î 🖬 🕺	3+					
🗾 /testbench/clk	1		սիսոսոստի	սոսոսոս				mmmm	mmmm		hhhhhhh	$\square$
/testbench/resetn	1 0									]		
/testbench/enable ⊕ /testbench/data	1 FFF8	0000							χeen (	0		
Image: A state of the state	000B	(0000 )		00000000		0000000				, 		
												_
	1000 n	s <b>Junio</b>	20	 0	4(		         		8		1	
	On	s <mark>Ons</mark>		-								
0 ns to 1050 ns												1.

## **Timing analysis**

Using the zoom controls and the cursors in the Wave window you can measure the length of time it takes for COUNT to change after a rising clock edge.

You will need to zoom in to an area in order to measure the time.

To make a time measurement between two edges, add cursors to the Wave window by clicking **Insert > Cursor** from the Wave menu. Drag one of the cursors to a rising edge of **clk** and drag the other cursor to the following transition on **count**. The difference between the cursors will be visible at the bottom of the Wave window. Note the time.



- 1. What was the time between the rising clock edge and COUNT changing?
- 2. What does this number represent?

Close ModelSim for Actelby clicking **File > Quit** from the main menu. Select **Yes** when prompted if you are sure you want to quit.

Close Libero IDE by clicking **File > Exit** from the main menu.

### **APPENDIX A: VHDL SOURCE CODE**

```
-- 16 bit loadable counter with enable and asynchronous reset
-- Actel Corporation
-- July 25, 2002
library ieee;
use ieee.std logic 1164.all;
use ieee.std logic unsigned.all;
entity counter16 is
port (CLK, RESETn, LOAD, ENABLE: in std logic;
        DATA: in std logic vector(15 downto 0);
        COUNT: out std logic vector(15 downto 0));
end counter16;
architecture RTL of counter16 is
signal COUNT_int: std_logic_vector(15 downto 0);
begin
process (RESETn, CLK)
begin
  if (RESETn = '0') then
      COUNT int <= (others => '0');
  elsif (CLK 'event and CLK = '1') then
      if (LOAD = '1') then
        COUNT int <= DATA;
      elsif (ENABLE = '1') then
        COUNT int <= COUNT int + 1;
      end if;
  end if;
end process;
COUNT <= COUNT int;
end RTL;
```

#### **Appendix B: VHLD Testbench**

```
-- Generated by WaveFormer Lite Version 8.9 at 17:44:57 on 10/25/2002
library ieee, std;
-- Libraries used by model under test
use IEEE.std logic 1164.all;
use IEEE.std logic unsigned.all;
-- End of libraries used by model under test
entity stimulus is
 port(
      CLK
                 : out std logic;
     RESETn
                        : out std logic;
     LOAD
                  : out std logic;
                        : out std logic;
     ENABLE
                  : out std logic vector(15 downto 0);
     DATA
                  : in std logic vector(15 downto 0)
     COUNT
      );
end stimulus;
architecture GeneratedCode of stimulus is
begin
CLK process : process
 variable tb stop time : time := 1201 ns;
  variable CLK Offset : time := 0 ns;
 variable CLK Period : time := 10 ns;
 variable CLK MinLH : time := 0 ns;
  variable CLK MaxLH : time := 0 ns;
  variable CLK MinHL : time := 0 ns;
  variable CLK MaxHL : time := 0 ns;
 variable CLK JRise : time := 0 ns;
 variable CLK JFall : time := 0 ns;
 variable CLK Duty : time := 50 ns;
 variable CLK high : time;
  variable CLK low : time;
begin
  CLK high := CLK Period * (CLK Duty / ns) / 100;
  CLK_low := CLK_Period - CLK_high;
  CLK <= '0';
  if (CLK Offset + (CLK MinLH - CLK JRise/2) < 0 ns) then
    assert FALSE report "Clock offset is less than 0 for CLK. This could be
caused by jitter. Increase offset to get rid of this error." severity
FAILURE;
  else
    wait for CLK Offset + (CLK MinLH - CLK JRise/2);
    while (now < tb stop time) loop
      CLK <= '1';
      wait for (CLK high - (CLK MaxLH + CLK JRise/2) + (CLK MinHL -
CLK JFall/2));
      CLK <= '0';
      wait for (CLK low - (CLK MaxHL + CLK JFall/2) + (CLK MinLH -
CLK JRise/2));
```

```
ACTEL TRAINING PROGRAM
VHDL LAB GUIDE FOR LIBERO IDE VER 2.3
    end loop;
 end if;
end process;
 process
 begin
       RESETn
               <=
                       '0';
       LOAD
                 <=
                       '0';
                       '0';
       ENABLE
                 <=
                       x"0000";
       DATA
                 <=
     wait for 35 ns;
                       -- Accumulated time = 35 ns
                       '1';
       RESETn
                 <=
     wait for 30 ns;
                       -- Accumulated time = 65 ns
       ENABLE
                 <=
                       '1';
     wait for 700 ns;
                      -- Accumulated time = 765 ns
       DATA
                 <=
                       x"FFF8";
     wait for 10 ns;
                       -- Accumulated time = 775 ns
       LOAD
                       '1';
                 <=
     wait for 30 ns;
                       -- Accumulated time = 805 ns
       LOAD
                      '0';
                 <=
     wait for 396 ns; -- Accumulated time = 1201 ns
     wait;
 end process;
end GeneratedCode;
--Top Level Test Bench Module
 -- Contains an instance of the Stimulus Module and an instance of the
Product Module
library ieee, std;
-- Libraries used by model under test
use IEEE.std logic 1164.all;
use IEEE.std_logic_unsigned.all;
-- End of libraries used by model under test
use work.all;
entity testbench is
 end testbench;
architecture tbGeneratedCode of testbench is
   -- Component declaration for the stimulus module
   component stimulus
   port(
     CLK
                 : out std_logic;
     RESETn
                       : out std logic;
     LOAD
                 : out std_logic;
     ENABLE
                       : out std_logic;
```

```
ACTEL TRAINING PROGRAM
VHDL LAB GUIDE FOR LIBERO IDE VER 2.3
     DATA
                 : out std logic vector(15 downto 0);
                 : in std_logic_vector(15 downto 0)
      COUNT
       );
   end component;
   -- Component declaration for the product module
   component counter
   port(
           : in std logic;
     CLK
     RESETn : in std logic;
     LOAD : in std logic;
     ENABLE : in std_logic;
     DATA : in std_logic_vector(15 downto 0);
     COUNT : out std logic vector(15 downto 0)
       );
   end component;
   -- Signal Declarations for the test bench module
   signal
                 CLK
                                   : std logic;
   signal
                 RESETn
                                         : std logic;
   signal
                LOAD
                                   : std logic;
   signal
                ENABLE
                                        : std logic;
   siqnal
                DATA
                                   : std logic vector(15 downto 0);
   signal
                 COUNT
                                   : std_logic_vector(15 downto 0);
   -- Ports are connected by matching the port names of the Test Module
  begin
    stimulus 0: stimulus
      port map(
                 CLK,
                 RESETn,
                 LOAD,
                 ENABLE,
                 DATA,
                 COUNT
                   );
    counter 0 : counter
     port map(
                 CLK => CLK,
                 RESETn => RESETn,
                 LOAD => LOAD,
                 ENABLE => ENABLE,
                 DATA => DATA,
                 COUNT => COUNT
                 );
end tbGeneratedCode;
```