Active-HDL LE Tutorial

Lattice Semiconductor Corporation
5555 NE Moore Court
Hillsboro, OR 97124
(503) 268-8000

April 2008

to correct any errors contained herein or to advise any user of this document of any correction if such be made. LSC recommends its customers obtain the latest version of the relevant information to establish, before ordering, that the information being relied upon is current.

Type Conventions Used in This Document

Convention	Meaning or Use
Bold	Items in the user interface that you select or click. Text that you type into the user interface.
<*Italic*>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
`Courier`	Code examples. Messages, reports, and prompts from the software.
`...`	Omitted material in a line of code.
. . .	Omitted lines in code and report examples.
`[ ]`	Optional items in syntax descriptions. In bus specifications, the brackets are required.
`( )`	Grouped items in syntax descriptions.
`{ }`	Repeatable items in syntax descriptions.
`|`	A choice between items in syntax descriptions.

Lattice® Semiconductor Corporation

Contents

Active-HDL LE Tutorial

This tutorial leads you through a typical simulation scenario using Active-HDL Lattice Edition (Active-HDL LE) as the simulation environment for ispLEVER. The tutorial design models a project using both VHDL and Verilog HDL blocks. It shows you how to set up ispLEVER and Active-HDL LE so you can use them interactively to manage a simulation run, use basic simulation and HDL debugging features, perform waveform analysis, and write scripts.

Although this tutorial emphasizes the feature set provided with Active-HDL LE, you can use all procedures described with more advanced versions of the Aldec Active-HDL product line.

Learning Objectives

When you have completed this tutorial, you should be able to do the following:

- Configure the ispLEVER Project Navigator to use Active-HDL LE as the default simulator.
- Execute simulation and perform HDL debugging tasks.
- Perform waveform analysis.
- Create scripts to automate simulation tasks.
- Use Active-HDL LE in both batch or interactive simulation scenarios.

This tutorial emphasizes using Active-HDL LE exclusively as a simulator launched as a “point” tool from the ispLEVER Project Navigator. Other Active-HDL licenses may include design entry features, such as schematics, state machines, and IP generators, and project management features that can be used to complement a Lattice Semiconductor FPGA design flow. Refer to the tutorial resources in the Aldec Active-HDL online help for more information.

Time to Complete This Tutorial

The time to complete this tutorial is about 90 minutes.

System Requirements

The following software configuration is required to complete the tutorial:

- Active-HDL Lattice Edition 7.3 or later
- ispLEVER-Starter (7.1 or later): FPGA Module and Precision RTL Synthesis Module

  or

  ispLEVER (7.1 or later) and Mentor Graphics® Precision® RTL Synthesis for Lattice

  or

  ispLEVER (7.1 or later) and Synplicity® Synplify Pro®

Note

Synplify for Lattice does not support concurrent synthesis of Verilog HDL and VHDL source.

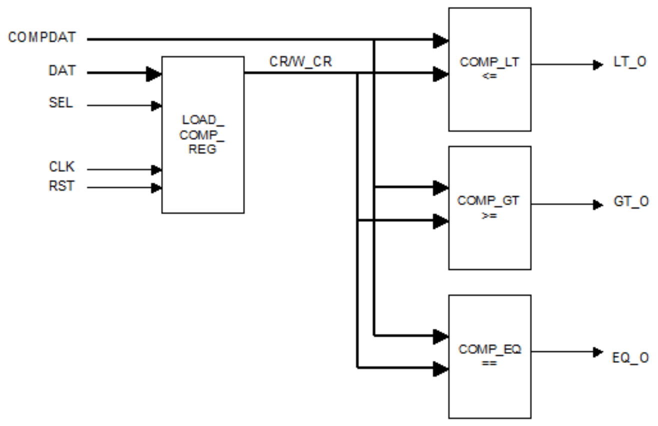
Accessing Online Help

You can find online help information on any tool included in the tutorial at any time by pressing the F1 key. Both ispLEVER Project Navigator and Active-HDL LE respond to F1 in a similar way.

About the Tutorial Design

The small design in this tutorial consists of both VHDL and Verilog HDL blocks targeted to a LatticeECP2M FPGA device. The block diagram in Figure 1 illustrates the design. Three comparator blocks evaluate the COMPDAT input data against the value of a storage register. Most of the model is coded in VHDL, and the COMP_EQ block is coded in Verilog.

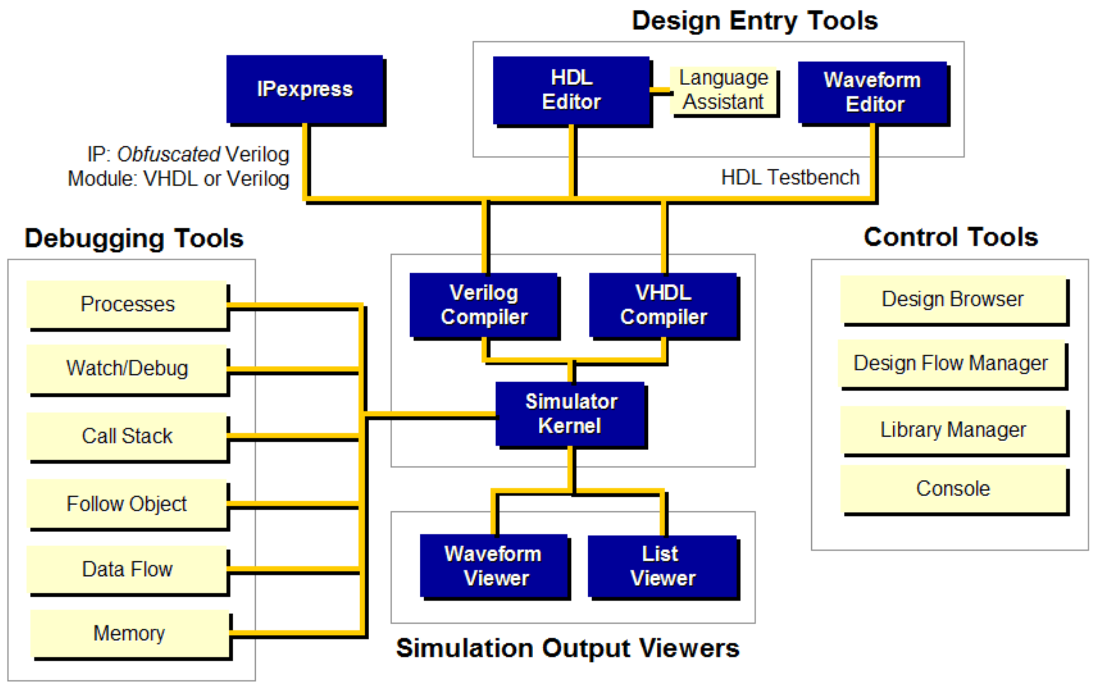
Figure 1: Tutorial Design



About Lattice Edition (LE)

Lattice Edition is a special configuration of Aldec Active-HDL that includes features focused on the needs of the FPGA designer, including mixed-language support, a variety of debugging tools, and powerful documentation and visualization tools. Figure 2 illustrates its architecture and major features.

Figure 2: Architecture and Features of Active-HDL LE



Task 1: Setting Up Project Navigator to Start Active-HDL

In the following procedure, you will configure the ispLEVER Project Navigator to use Active-HDL LE as the default HDL simulator. Whenever a simulation-related process is run from within Project Navigator, Active-HDL LE is launched with a script to compile the simulation file list and run a default simulation based on the test bench.

To configure Project Navigator to start Active-HDL:

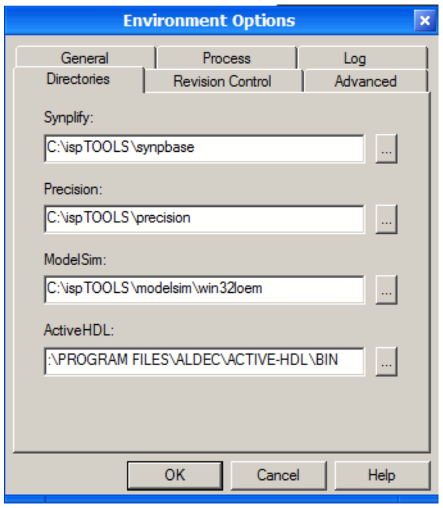
1. Start the ispLEVER Project Navigator, if it is not already running.
2. From Project Navigator, choose **Options > Environment**.

   The Environment Options dialog box appears.
3. Click the **Directories** tab, then click the browse (**...**) button next to the Active-HDL box.

   The Set Active-HDL Install Path dialog box appears.
4. Select the folder where the Active-HDL LE executable file (ahdl.exe) is installed, for example, c:\Program Files\Aldec\Active-Hdl <*version*>\bin.

The directory for Active-HDL LE appears in the ActiveHDL box in the Directories tab of the Environment Options dialog box, as shown in Figure 3.

Figure 3: Active-HDL LE Directory in Environment Options Dialog Box



Synplify: C:\ispTOOLS\synpbase

Precision: C:\ispTOOLS\precision

ModelSim: C:\ispTOOLS\modelsim\win32loem

ActiveHDL: :\PROGRAM FILES\ALDEC\ACTIVE-HDL\BIN

5. Click **OK**.

   The Active-HDL LE button appears in the Project Navigator toolbar.

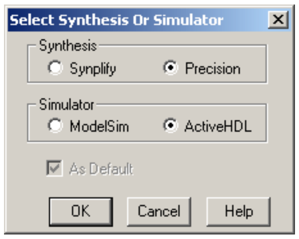
6. Choose **Options > Select RTL Synthesis or Simulator**.

7. In the Synthesis box of the Select Synthesis or Simulator dialog box, choose **Precision**.

8. In the Simulator box, choose **ActiveHDL**.

   Figure 4 illustrates these settings.

Figure 4: Select Synthesis Or Simulator Dialog Box



Synthesis: Synplify [✓ Precision]

Simulator: ModelSim [✓ ActiveHDL]

✓ As Default

9. Click **OK**.

Task 2: Create a New Project

The ispLEVER software employs the concept of a project. A project is a design. Each project has its own directory in which all source files, intermediate data files, and resulting files are stored. To begin a new project, you must create a project directory. Then you must give the project file a name with a .syn suffix. The Project Navigator will use this file name later to reload the project.

Since this tutorial incorporates a combination of VHDL and Verilog source files, a mixed-language-type project is required.

Note

If you want to preserve the original tutorial design files, save the active_hdl_tutor directory to another location on your computer before proceeding.

To create a new project:

1. Start the ispLEVER Project Navigator, if it is not already running.
2. In the Project Navigator, choose **File > New Project**.
3. In the Project Wizard dialog box, enter or select the following:
   a. In the Project Name box, enter **compare**.
   b. In the Location box, enter the following directory:

      ***<install_path>*\examples\Tutorial\active_hdl_tutor**
   c. In the Design Entry Type box, select **Mixed Verilog/VHDL**.
   d. In the Synthesis Tools box, choose **Precision** (or **Synplify** if you have access to a Synplify Pro license).
   e. In the Simulator Tools box, choose **ActiveHDL**.
4. Click **Next**.
5. In the Project Wizard – Select Device dialog box, enter or select the following:
   a. In the Family box, choose **LatticeECP2**.
   b. In the Device box, choose **LFE2-6E**.
   c. In the Speed Grade box, choose **-5**.
   d. In the Package type box, choose **TQFP144**.
   e. In the Operating Conditions box, choose **Commercial**.
6. Click **Next**.
7. In the Project Wizard – Add Source dialog box, click **Add Source**. appears.
8. In the Import File: (Mixed Verilog/VHDL) dialog box, select the following HDL files:
   - comp_eq.v

- ◆ gt.vhd
- ◆ lt.vhd
- ◆ tb_top.vhd
- ◆ top.vhd

9. Click **Open**.

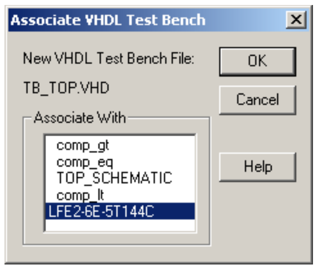
   In the Import Source Type dialog box, choose the following source type using the following pattern:

comp_eq.v	Verilog Module
lt.vhd	VHDL Module
gt.vhd	VHDL Module
tb_top.vhd	VHDL Test Bench
top.vhd	VHDL Module

   The file source types tell the Project Navigator how to construct file lists for logic synthesis and simulation. In this case, the VHDL Test Bench, tb_top.vhd, is included in the file list for simulation but is ignored during the synthesis process.

10. Click **Next**.
11. In the The Project Wizard – Project Information dialog box, click **Finish**.
12. In the Associated VHDL Test Bench dialog box, shown in Figure 5, choose **LFE2-6E-5T144C** and click **OK**.

Figure 5: Associated VHDL Test Bench Dialog box


Associate VHDL Test Bench

New VHDL Test Bench File:

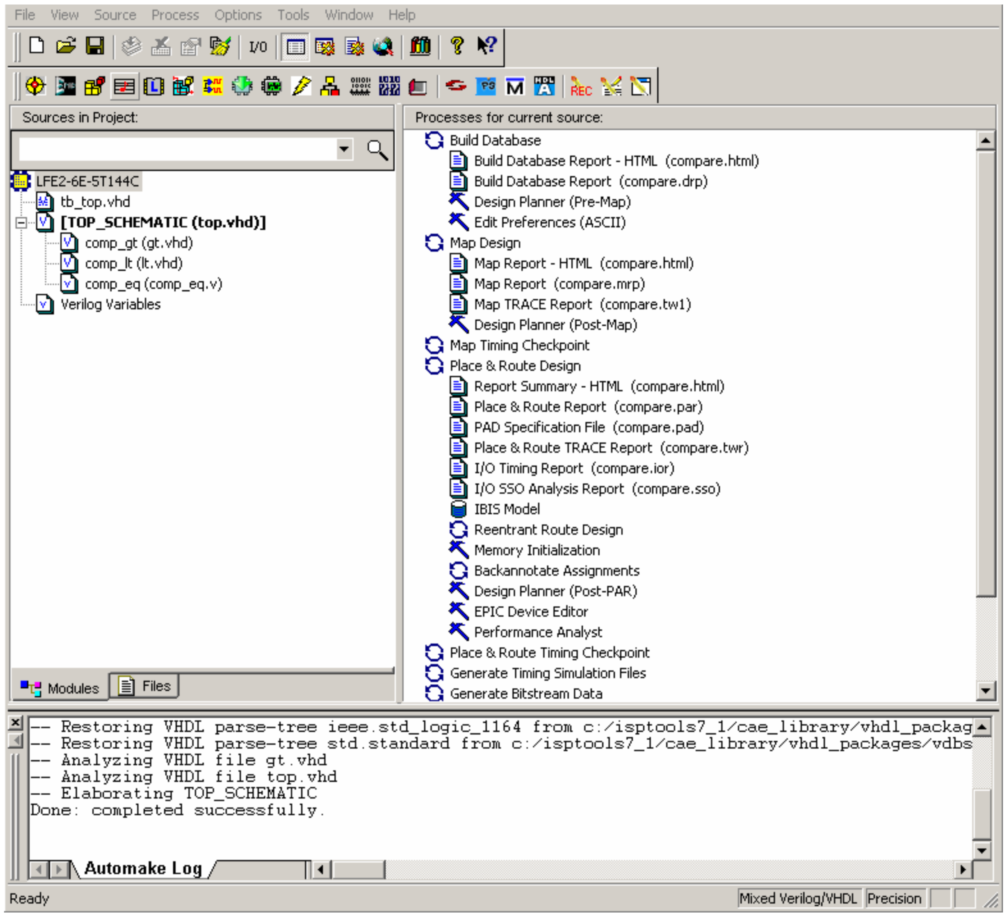
TB_TOP.VHD

Associate With

comp_gt
comp_eq
TOP_SCHEMATIC
comp_lt
LFE2-6E-5T144C

The Project Navigator analyzes all HDL source files and creates a list of modules in the Modules tab of the Sources in Project window, as shown in Figure 6. The VHDL test bench file, tb_top.vhd, appears with a special icon above top.vhd.

Figure 6: Modules in Modules Tab of Project Navigator


File View Source Process Options Tools Window Help
Sources in Project:
Processes for current source:
Modules
Files
Automake Log
Ready
Mixed Verilog/VHDL
Precision

Sources in Project:

- LFE2-6E-5T144C
  - tb_top.vhd
  - [TOP_SCHEMATIC (top.vhd)]
    - comp_gt (gt.vhd)
    - comp_lt (lt.vhd)
    - comp_eq (comp_eq.v)
  - Verilog Variables

Processes for current source:

- Build Database
  - Build Database Report - HTML (compare.html)
  - Build Database Report (compare.drp)
  - Design Planner (Pre-Map)
  - Edit Preferences (ASCII)
- Map Design
  - Map Report - HTML (compare.html)
  - Map Report (compare.mrp)
  - Map TRACE Report (compare.tw1)
  - Design Planner (Post-Map)
- Map Timing Checkpoint
- Place & Route Design
  - Report Summary - HTML (compare.html)
  - Place & Route Report (compare.par)
  - PAD Specification File (compare.pad)
  - Place & Route TRACE Report (compare.twr)
  - I/O Timing Report (compare.ior)
  - I/O SSO Analysis Report (compare.sso)
  - IBIS Model
  - Reentrant Route Design
  - Memory Initialization
  - Backannotate Assignments
  - Design Planner (Post-PAR)
  - EPIC Device Editor
  - Performance Analyst
- Place & Route Timing Checkpoint
- Generate Timing Simulation Files
- Generate Bitstream Data

```
-- Restoring VHDL parse-tree ieee.std_logic_1164 from c:/isptools7_1/cae_library/vhdl_packag
-- Restoring VHDL parse-tree std.standard from c:/isptools7_1/cae_library/vhdl_packages/vdbs
-- Analyzing VHDL file gt.vhd
-- Analyzing VHDL file top.vhd
-- Elaborating TOP_SCHEMATIC
Done: completed successfully.
```

Task 3: Functional Simulation

Functional simulation in the ispLEVER design flow allows you to confirm the behavior of the project model without regard to the timing characteristics of the target device. Delay through the model is either single-unit (1 ps) delay or zero, depending on how the model is constructed. Source files at this stage are typically the register transfer level (RTL) HDL, IP cores, and a test bench to create input stimulus patterns and confirm that the output logic is correct. You may also request a gate-level, post-route functional simulation that reflects the post-synthesis, post-PAR version of the design.

You can access functional simulation in one of two ways from the Project Navigator:

- Batch interface, which starts Active-HDL LE using a Tcl script that references the project file list, libraries for the target FPGA, and a default waveform display and run time. No simulation project is created. The batch interface is preferred when the file list is not yet stable and when there is no need for a simulation project to carry options from session to session.

- Interactive interface, which starts Active-HDL LE from the Tools menu (or toolbar) or Start menu. You can create a simulation project and manage the file list and libraries for the target FPGA manually. You can also run and debug the design interactively. The interactive interface is preferred when the project source is more stable and you require intimate control over the simulator that the batch interface does not provide.

This task demonstrates the batch simulation interface. For more information on using Active-HDL LE interactively, see the Active-HDL online help and tutorials.

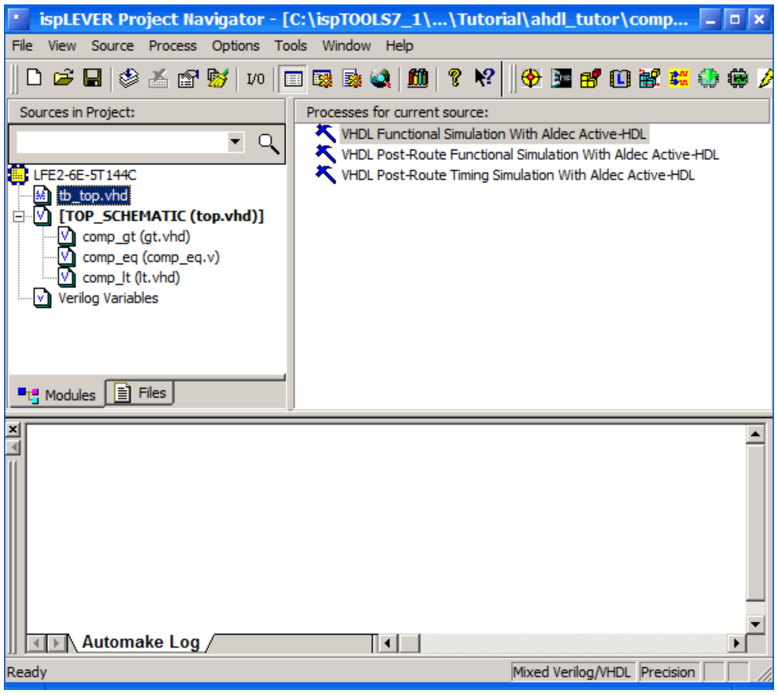
To run the batch interface to Active-HDL LE:

1. From the Sources in Project window, select **tb_top.vhd**.

   The Processes for current source window adjusts to display three simulation processes, as shown in Figure 7:

   - VHDL Functional Simulation with Aldec Active-HDL
   - VHDL Post-Route Functional Simulation with Aldec Active-HDL
   - VHDL Post-Route Timing Simulation with Aldec Active-HDL

Figure 7: Simulation Processes in Project Navigator


ispLEVER Project Navigator - [C:\ispTOOLS7_1\...\Tutorial\ahdl_tutor\comp...

Sources in Project:
LFE2-6E-5T144C
tb_top.vhd
[TOP_SCHEMATIC (top.vhd)]
comp_gt (gt.vhd)
comp_eq (comp_eq.v)
comp_lt (lt.vhd)
Verilog Variables

Processes for current source:
VHDL Functional Simulation With Aldec Active-HDL
VHDL Post-Route Functional Simulation With Aldec Active-HDL
VHDL Post-Route Timing Simulation With Aldec Active-HDL

The tutorial project could have used a Verilog test fixture for stimulus as well. You can use any combination of HDL types within the project, as long as your simulator and synthesis tool is licensed to support mixed HDL.

To run gate-level simulation or gate-level-with-timing simulation based on post-place-and-route data, use either of the Post-Route processes that appear when you select a test bench file.

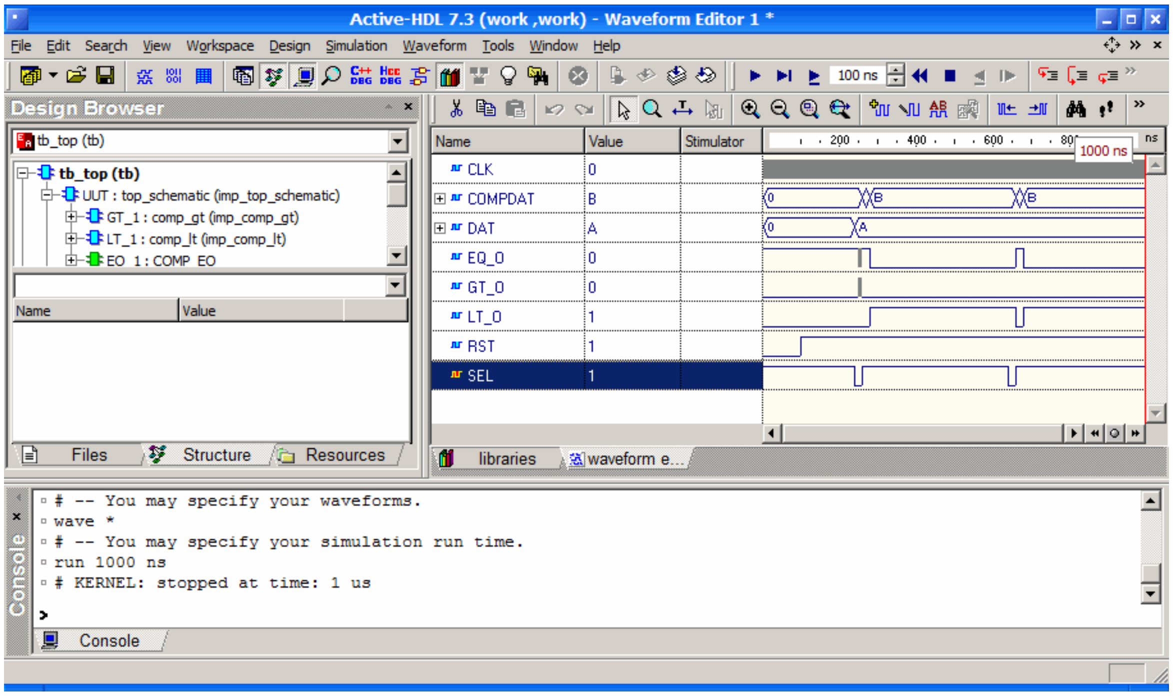
2. Double-click the **VHDL Functional Simulation with Aldec Active-HDL** process.

   The Active-HDL LE interface appears and runs the batch script produced by Project Navigator. Waveform Editor 1 appears with signal patterns based on the test bench and output of the model.

3. Click the **Zoom to Fit** button of the Waveform Editor toolbar.

   The waveform adjusts so all events are visible in the display, as shown in Figure 8.

Figure 8: Adjusting the Waveform Display


Active-HDL 7.3 (work ,work) - Waveform Editor 1 *
File Edit Search View Workspace Design Simulation Waveform Tools Window Help
Design Browser
tb_top (tb)
tb_top (tb)
UUT : top_schematic (imp_top_schematic)
GT_1 : comp_gt (imp_comp_gt)
LT_1 : comp_lt (imp_comp_lt)
EQ_1 : COMP_EQ
Name
Value
Files
Structure
Resources
Name
Value
Stimulator
CLK 0
COMPDAT B
DAT A
EQ_O 0
GT_O 0
LT_O 1
RST 1
SEL 1
200
400
600
1000 ns
libraries
waveform e...
Console

```
# -- You may specify your waveforms.
wave *
# -- You may specify your simulation run time.
run 1000 ns
# KERNEL: stopped at time: 1 us
>
```

4. Scroll up through the Console tab, noting the Tcl commands issued by the batch interface.

Following is the entire sequence of scripts:

`tb_top_activehdl.do:`
```
set SIM_WORKING_FOLDER .
do -tcl tb_top.fado
```

`tb_top.fado:`
```
# NOTE: Do not edit this file.
# Auto generated by VHDL Functional Simulation Models
#
design create work ./work
design open work/work
do -tcl compare.vfd
```

```
vcom  tb_top.vhd
vsim TB_TOP -L pmi_work
do tb_top_vhdf.udo TB_TOP
# End
```

compare.vfd:

```
# NOTE: Do not edit this file.
# Auto-generated by All VHDL Functional Simulation Models
#
source {C:/ispTOOLS7_1/ispcpld/bin/chipsim_cmd.tcl}
set sty_file compare.sty
if {![info exists HasProc(LS_Vcom)]} { source {chipsim_cmd.tcl}
}
set vcom_opt ""
set src_files {comp_eq.v lt.vhd gt.vhd top.vhd}
LS_Vcom sty_file src_files vcom_opt
# End
```

tb_top_vhdlf.udo:

```
-- ispLEVER VHDL Functional Simulation Template:
tb_top_vhdf.udo.
-- You may edit this file to control your simulation.
-- You may specify your design unit. - 1
-- You may specify your waveforms.
add wave *
-- You may specify your simulation run time.
run 1000 ns
```

The details of all the scripts are not important to understand; however, the tb_top_vhdlf.udo script provides a means to specify the signals to be traced in the waveform window and default simulation run time. This is a popular file to modify for your project.

Task 4: Browsing the Design

Many of the debugging and analysis tasks you will perform in the simulator are based on the simulation objects that you select from the Design Browser. Simulation objects refer to elements such as component instances, signals, ports, and variables of the model.

The Structure tab of the Design Browser provides a browser that presents the hierarchical organization of the model. Different icons are used to indicate component instances, signal drivers, and processes. Icons are color-coded to indicate the source format as VHDL or Verilog HDL. As you select objects of the structure, a list of ports, signals, and variable appears below the tree.

Selections that you make in the Design Browser are used to populate a variety of visualization features, such as the Waveform, Watch, and List displays.

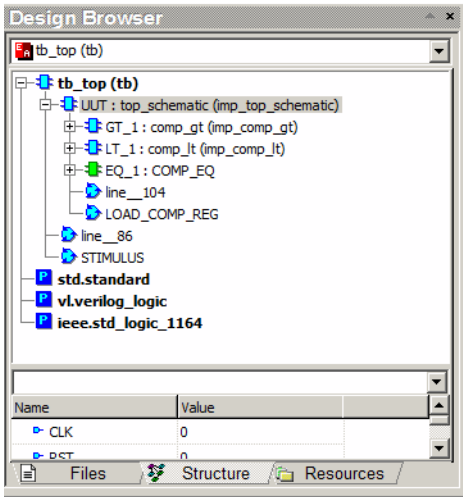
To browse the design:

1. From the Design Browser, click the [+] symbol next to **tb_top (tb)**, then click the [+] symbol next to **UUT: top_schematic (imp_top_schematic)** branch.

The details of the instance appear, and the port and signal contents appear in the list below, as shown in Figure 9.

The VHDL objects of the model are colored blue, and the Verilog objects are colored green.

Figure 9: Design Browser



Task 5: Source-Level Debugging

The source-level debugging features of the simulator enable you to stop the simulation run if certain signal conditions are met or when certain lines of a source file are executed. Stopping the simulation enables you to the examine states of the model as it executes to help detect bugs that lead to incorrect function.

Active-HDL LE provides two breakpoint types: code and signal. A code breakpoint suspends the simulation run when the simulator encounters an executable code statement where a breakpoint is set. A signal breakpoint suspends the simulation run when a model's signal matches a user-defined value.

Controlling the Simulation with a Code Breakpoint

You will insert a breakpoint into the code to suspend the simulation.

To control the simulation run with a code breakpoint:

1. Choose **Simulation > Restart Simulation**.

   The simulation is initialized to time zero (0 ps).

2. Select the **Structure** tab of the Design Browser window, if it is not already selected.

3. Double-click the test bench module, **tb_top (tb)**.

The tb_top.vhd file appears in the HDL editor window.

4. Double-click the top-level HDL module instance, **UUT: top_schematic (imp_top_schematic)**.

   The top.vhd file appears in the HDL editor window.

5. Scroll to the LOAD_COMP_REG process and reset expression (**`if RST = '0' then`**) near line 115.

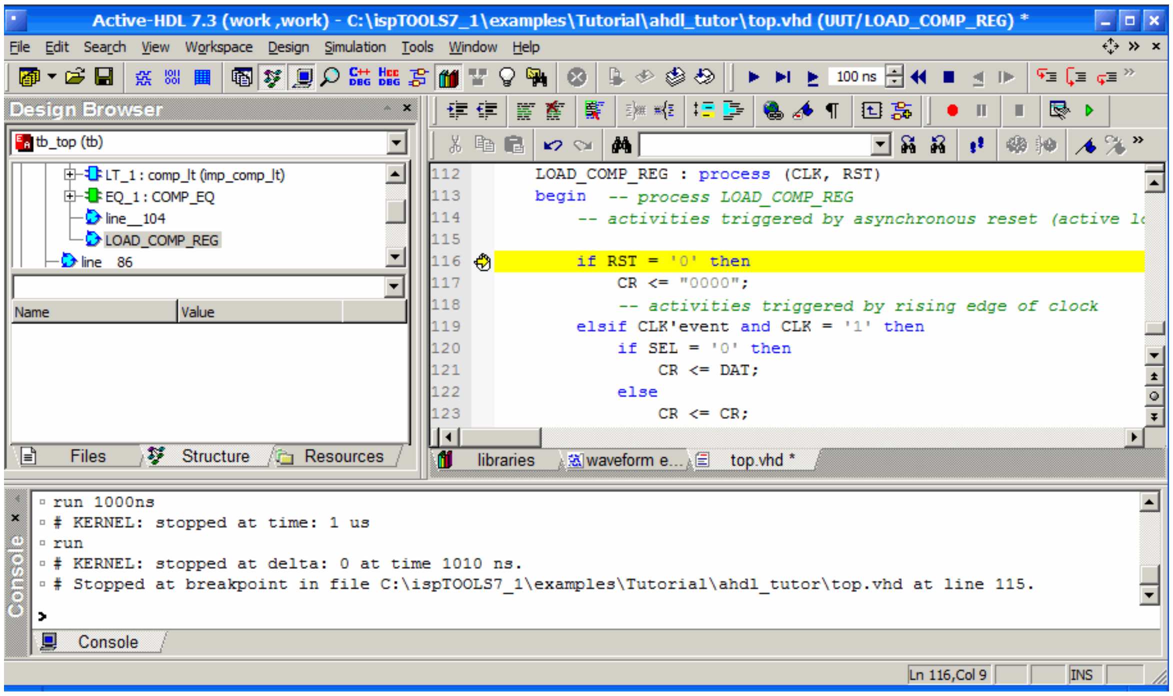
6. Press **F9** or click the right mouse button on the reset expression and choose **Toggle Breakpoint**.

   A breakpoint icon appears in the left column of the HDL editor.

7. Press **Alt+F5** or choose **Simulation > Run**.

   The simulator executes the model until the reset expression is examined. The line of source with the active breakpoint is highlighted, as shown in Figure 10.

Figure 10: Simulation Stopped at Breakpoint


Active-HDL 7.3 (work ,work) - C:\ispTOOLS7_1\examples\Tutorial\ahdl_tutor\top.vhd (UUT/LOAD_COMP_REG) *
File Edit Search View Workspace Design Simulation Tools Window Help
100 ns
Design Browser
tb_top (tb)
LT_1 : comp_lt (imp_comp_lt)
EQ_1 : COMP_EQ
line__104
LOAD_COMP_REG
line__86
Name
Value
Files
Structure
Resources
libraries
waveform e...
top.vhd *
Console
Ln 116,Col 9
INS

```
112        LOAD_COMP_REG : process (CLK, RST)
113        begin  -- process LOAD_COMP_REG
114            -- activities triggered by asynchronous reset (active lo
115
116            if RST = '0' then
117                CR <= "0000";
118                -- activities triggered by rising edge of clock
119            elsif CLK'event and CLK = '1' then
120                if SEL = '0' then
121                    CR <= DAT;
122                else
123                    CR <= CR;
```

```
run 1000ns
# KERNEL: stopped at time: 1 us
run
# KERNEL: stopped at delta: 0 at time 1010 ns.
# Stopped at breakpoint in file C:\ispTOOLS7_1\examples\Tutorial\ahdl_tutor\top.vhd at line 115.
>
```

Once the simulator pauses at a breakpoint, you can advance the simulator by using one of four trace options. The trace option that you use depends on what language constructs you wish to examine.

- Trace Into – Executes a single statement. If a subprogram call is encountered, the execution descends into the subprogram body.
- Trace Over – Executes a single statement. If a subprogram call is encountered, the statements of the subprogram body are executed in a single step.

- Trace Out – Executes as many statements as required to complete the execution of a subprogram.
- Trace Over Transition – Executes as many statements as are required to perform a transition between states (state machines only).

8. Press **F7** (Trace Into) to step the simulator.

   This step continues simulation execution, tracing coding line by line. A yellow highlight steps through the source as the model executes.

   In some cases, your breakpoint may not be triggered as you expect and the simulator will run indefinitely, as long as events are generated by your model. To interrupt the simulator, press the **Break** button of the toolbar.

   Note the difference between Break (toolbar) and End Simulation (Simulation toolbar) features:

   - The Break feature interrupts the execution of a run. A signal history recorded in the Waveform or List views is available to you for analysis, and you can continue simulating from the time you requested the break.
   - The End Simulation feature both interrupts the simulator and clears the memory. After End Simulation, you must reinitialize the simulator and rerun to establish the in-memory model.

9. Select various branches of the Design Browser window.

   Port and signal states appear in the lower panel of the window. This feature allows you to examine the state of a port or signal anywhere in the model, even if you did not initially trace it in the waveform window.

10. Choose **Simulation > Breakpoints**.
11. In the Breakpoints dialog box, disable the code breakpoint by clearing the check box in the leftmost column. Keep the dialog box open.

Controlling the Simulation with a Signal Breakpoint

Now you will add a signal breakpoint to suspend the simulation run.

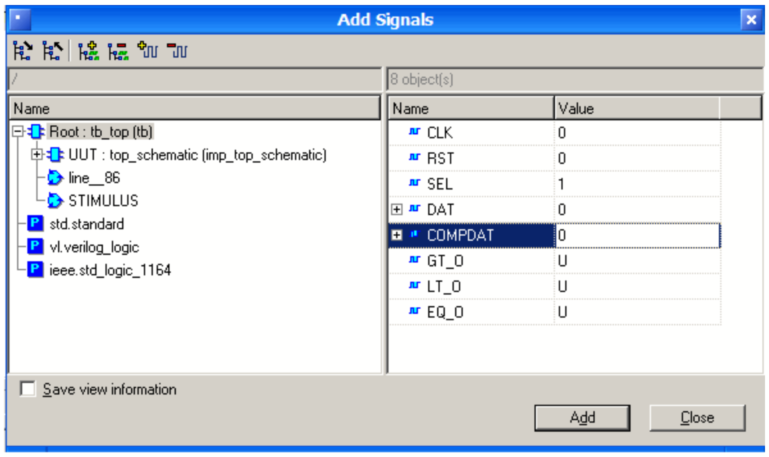
To control the simulation run with a signal breakpoint:

1. Click the **Signal Breakpoints** tab in the Breakpoints dialog box.
2. Click the **Add Signals** button.

   The Add Signals dialog box appears. A display similar to the Design Browser appears. On the left is a tree of the model hierarchy. On the right is a display of ports and signals at the current level of hierarchy.

3. Choose **Root: tb_top (tb)** from the hierarchy browser, then select the **COMPDAT** signal, as shown in Figure 11.

Figure 11: Choosing the COMPDAT Signal in the Add Signals Dialog Box



Name	Value
CLK	0
RST	0
SEL	1
DAT	0
COMPDAT	0
GT_0	U
LT_0	U
EQ_0	U

4. Click the **Add** button.

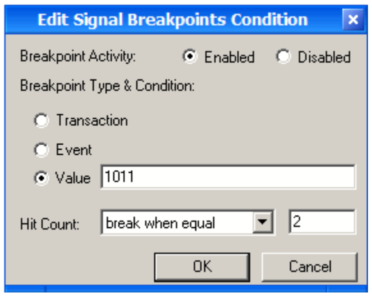
   COMPDAT is added to the Signal Breakpoints list in the Breakpoints dialog box.

5. Click the **Close** button in the Add Signals dialog box.
6. Click **Edit Condition**.
7. In the Edit Signal Breakpoints Condition dialog box, select the following options.

   a. In the Value box, enter **1011**.

   b. In the Hit Count box, choose **break when equal** from the drop-down menu, as shown in Figure 12. In the box to the right of Hit Count, enter **2**.

Figure 12: Setting the Breakpoint in the Edit Signal Breakpoints Condition Dialog box



Breakpoint Activity: [✓ Enabled] Disabled

Breakpoint Type & Condition:

Transaction

Event

[✓ Value] 1011

Hit Count: break when equal 2

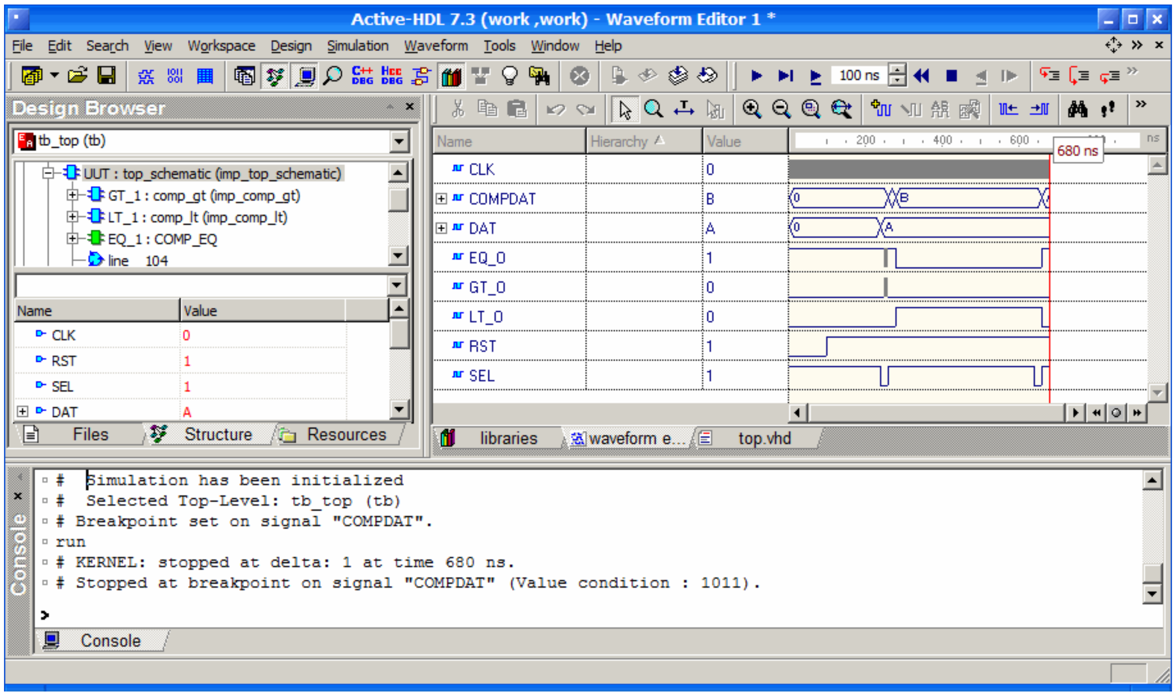
This breakpoint condition triggers the second time COMPDAT reaches the state 1011b (Bh).

c. Click **OK**.

8. Click **OK** in the Breakpoints dialog box.
9. Choose **Simulation > Restart Simulation** to re-initialize the simulator.
10. Press **Alt+F5** or the **Run** button on the toolbar.

    The simulator runs until the COMPDAT signal breakpoint occurs around time 680 ns, as shown in Figure 13 (you may need to click on the Waveform Editor 1 tab to view it).

Figure 13: Reaching the COMPDAT Signal Breakpoint


Active-HDL 7.3 (work ,work) - Waveform Editor 1 *
File Edit Search View Workspace Design Simulation Waveform Tools Window Help
100 ns
Design Browser
tb_top (tb)
UUT : top_schematic (imp_top_schematic)
GT_1 : comp_gt (imp_comp_gt)
LT_1 : comp_lt (imp_comp_lt)
EQ_1 : COMP_EQ
line 104
Files
Structure
Resources
200
400
600
680 ns
ns
libraries
waveform e...
top.vhd
Console

Name	Value
CLK	0
RST	1
SEL	1
DAT	A

Name	Hierarchy	Value
CLK		0
COMPDAT		B
DAT		A
EQ_O		1
GT_O		0
LT_O		0
RST		1
SEL		1

```
# Simulation has been initialized
#  Selected Top-Level: tb_top (tb)
# Breakpoint set on signal "COMPDAT".
run
# KERNEL: stopped at delta: 1 at time 680 ns.
# Stopped at breakpoint on signal "COMPDAT" (Value condition : 1011).
>
```

Viewing Instantaneous States

Once the execution of the simulation has stopped, you can view the instantaneous states of a variable, port, or signal of the model to detect factors causing malfunction.

To view the instantaneous states of a variable, port, or signal:

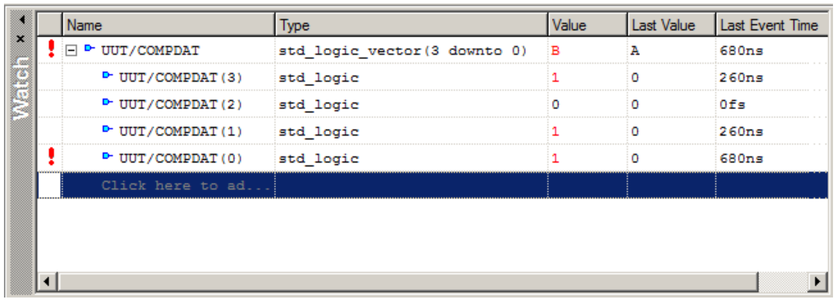
1. Choose **View > Watch**.

   The Watch window appears.

2. From the Design Browser window, select **UUT : top_schematic (imp_top_schematic)**.
3. From the Name/Value list of the Design Browser, select **COMPDAT**, then drag and drop it to the Watch window.

Details about the object and current value appear in the display, as shown in Figure 14.

Figure 14: Information on COMPDAT Signal Displayed in Watch Window



Name	Type	Value	Last Value	Last Event Time
UUT/COMPDAT	std_logic_vector(3 downto 0)	B	A	680ns
UUT/COMPDAT(3)	std_logic	1	0	260ns
UUT/COMPDAT(2)	std_logic	0	0	0fs
UUT/COMPDAT(1)	std_logic	1	0	260ns
UUT/COMPDAT(0)	std_logic	1	0	680ns
Click here to ad...				

Task 6: Waveform Analysis

The waveform display is the most popular means of examining the history of signal and port objects in the simulation model. Several visualization and measurement features are provided to help you better understand the operation of the model and the relationship between signals.

Adding Port and Signal Objects

You can add port and signal objects to the Waveform Editor.

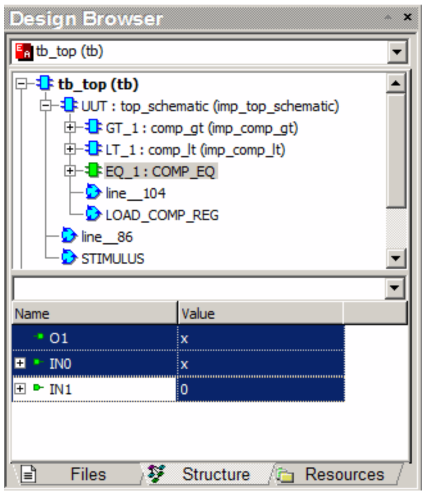
To add port and signal objects to the Waveform Editor:

1. In the Design Browser, select **EQ_1: COMP_EQ**.

   The EQ_1 ports appear in the list below.

2. Select ports **O1**, **IN0**, and **IN1**, as shown in Figure 15, right-click, then select **Add to Waveform**.

Figure 15: Ports Selected in Design Browser



Design Browser

tb_top (tb)

- tb_top (tb)
  - UUT : top_schematic (imp_top_schematic)
    - GT_1 : comp_gt (imp_comp_gt)
    - LT_1 : comp_lt (imp_comp_lt)
    - EQ_1 : COMP_EQ
    - line__104
    - LOAD_COMP_REG
  - line__86
  - STIMULUS

Name	Value
O1	x
IN0	x
IN1	0

The ports are added to the Waveform Editor. You can also drag and drop simulation objects between many other windows and panes.

3. Right-click the **Name** column header of the waveform view and select **Columns > Hierarchy**.

   A new column appears to indicate the hierarchical path (UUT/EQ_1) to the new ports.

4. In the Design Browser, select the **LT_1** instance, then drag and drop it onto the waveform window.

   Three LT_1 ports appear in the waveform window.

   In the following final steps, you will add signals using an Active-HDL Tcl command.

5. In the Design Browser, select instance **UUT**.

   The UUT signals and ports appear in the list below the hierarchy.

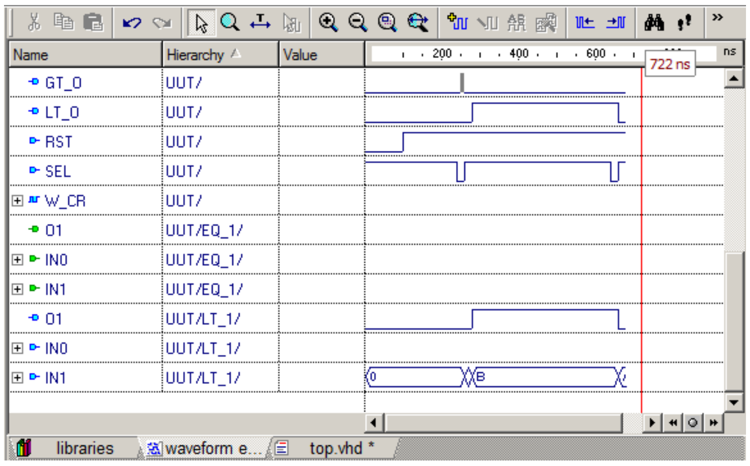
6. In the Console window, type **wave *** at the command prompt and press **Enter**.

   All UUT ports and signals appear in the waveform window.

7. In the Console window, type **run** and press **Enter**.

   The simulator advances, and the waveform display shows the state history of all ports and signals traced, as shown in Figure 16.

Figure 16: State History of Traced Ports and Signals



Name	Hierarchy	Value
GT_0	UUT/	
LT_0	UUT/	
RST	UUT/	
SEL	UUT/	
W_CR	UUT/	
01	UUT/EQ_1/	
IN0	UUT/EQ_1/	
IN1	UUT/EQ_1/	
01	UUT/LT_1/	
IN0	UUT/LT_1/	
IN1	UUT/LT_1/	

Adjusting the Waveform View

You can adjust the scale of the waveforms in the Waveform Editor.

To adjust the waveform view:

1. Click the **>>** icon of the menu bar to detach Waveform Editor 1.

   The waveform appears in a stand-alone window.

2. Click the **Zoom To Fit** button on the Waveform Editor toolbar.

The waveform scale changes to fill the window.

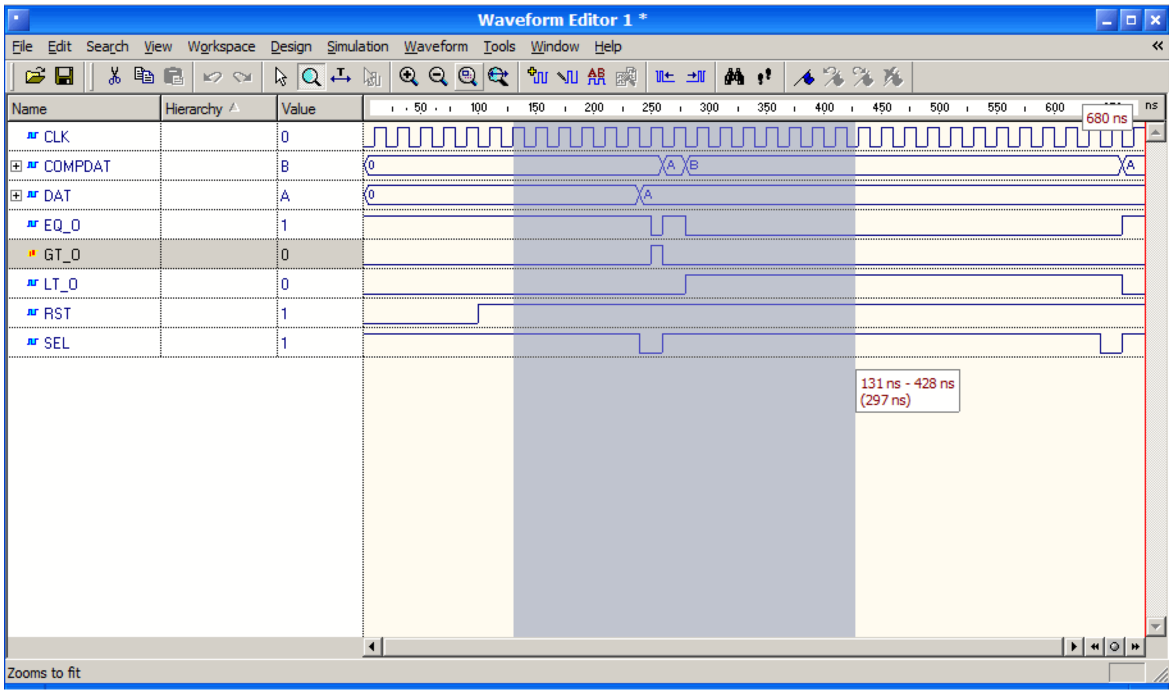
3. Right-click on the waveform and choose **Zoom Mode** from the pop-up menu.

   The cursor shape changes from Select Mode to a magnifying-glass shape.

4. Select a portion of the waveform by pressing down the left mouse button, dragging the mouse over the period approximately **100 ns-400 ns**, and releasing the button. Refer to the Waveform Viewer time scale to select the period.

   The waveform scale changes to fill the view with the selected period, as shown in Figure 17.

Figure 17: Changing the Waveform Scale



5. Click the **Zoom To Fit** button on the Waveform Editor toolbar.

   The waveform scale changes to fill the window.

Measuring the Time Between Waveform Events

The next procedure demonstrates how to measure the active period of the GT_O output port. You can do this by positioning cursors or by using Measurement Mode.

Positioning Cursors to Measure Time Between Waveform Events

To measure the time between waveform events by positioning cursors:

1. Right-click on the waveform, and choose **Select Mode**.

   The Zoom Mode is deactivated.

2. Right-click on the waveform, and choose **Add Cursor**.

   A red cursor appears on the display and the time delta between it and the end of the simulation.

3. Select the **GT_O** signal.
4. Select the cursor and drag it to any point in the time line before 250 ns.
5. Press the **F12** or the **Next Event** button of the Waveform Editor toolbar.

   The cursor snaps to the first rising transition of GT_O.

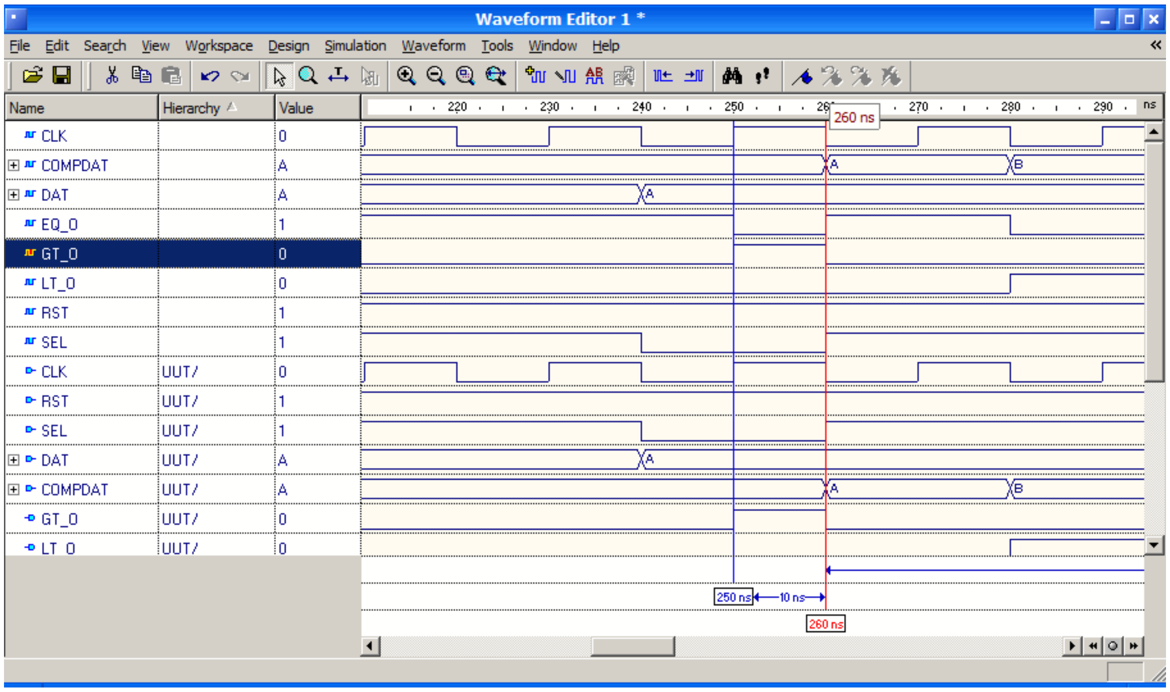
6. Right-click on the waveform and choose **Add Cursor**.

   Another cursor appears on the display.

7. Select the new cursor and drag it to any point in the time line after 300 ns.
8. Press the **Previous Event** button of the waveform view toolbar.

The second cursor snaps to the falling transition of GT_O. The pulse period is displayed as 10 ns, as shown in Figure 18.

Figure 18: Pulse Period



Using Measurement Mode to Measure Time Between Waveform Events

An alternative to positioning cursors to determine a time delta is to use the Measurement Mode feature to annotate delta and markers between signal events.

To measure the time between waveform events by using Measurement Mode:

1. From the waveform display, right-click and choose **Measurement Mode**.

   The cursor changes to the measurement icon.

2. Click and hold the left mouse button at the transition from A to B of the COMPDAT signal, then drag the cursor to the right. Release the mouse at an edge of the CLK signal.

   A measurement tag is annotated onto the waveform.

   You can also create a temporary tag by holding down the Ctrl key when you measure between events. Click on the display to clear the period measurement.

3. Click the **<<** icon of the menu bar.

   The Waveform Editor is reattached to the Active-HDL LE frame.

4. Choose **Simulation > Restart Simulation** or click the Restart Simulation button of the simulation toolbar.

The simulation is initialized.

Task 7: Scripting a Simulation

In many cases, command scripts are faster to execute than the equivalent sequence of actions in the graphical user interface. Active-HDL LE supports batch operation through macro files that you can run from the Console window of the Active-HDL user interface.

For more information on macro commands, see the *Active-HDL Macro Language* section of the Active-HDL LE online help.

Note

Advanced versions of Active-HDL provide a stand-alone simulation environment, VSimSA, that is designed for long simulation runs and batch processing from the Windows command line. The Lattice Design Edition Lite license does not support VSimSA.

To create and run an Active-HDL macro file:

1. Choose **File > New > Macro.**

   A new text editor tab appears.

2. Create the following macro script:

```
# Macro1.do

# Create a library
alib work
# Compile the simulation file list
vlog -v2k comp_eq.v
vcom -2002 lt.vhd gt.vhd top.vhd tb_top.vhd
# Initialize
vsim TB_TOP -L pmi_work
# Save simulation data to a VCD text file
vcd file wave.vcd
vcd add -r TB_TOP *
# List all signals in decimal format
list *
wave *
# Advance the simulator 700 ns
run 700 ns
# Quit the simulation
endsim
# End
```

3. Save the file as **macro1.do**.

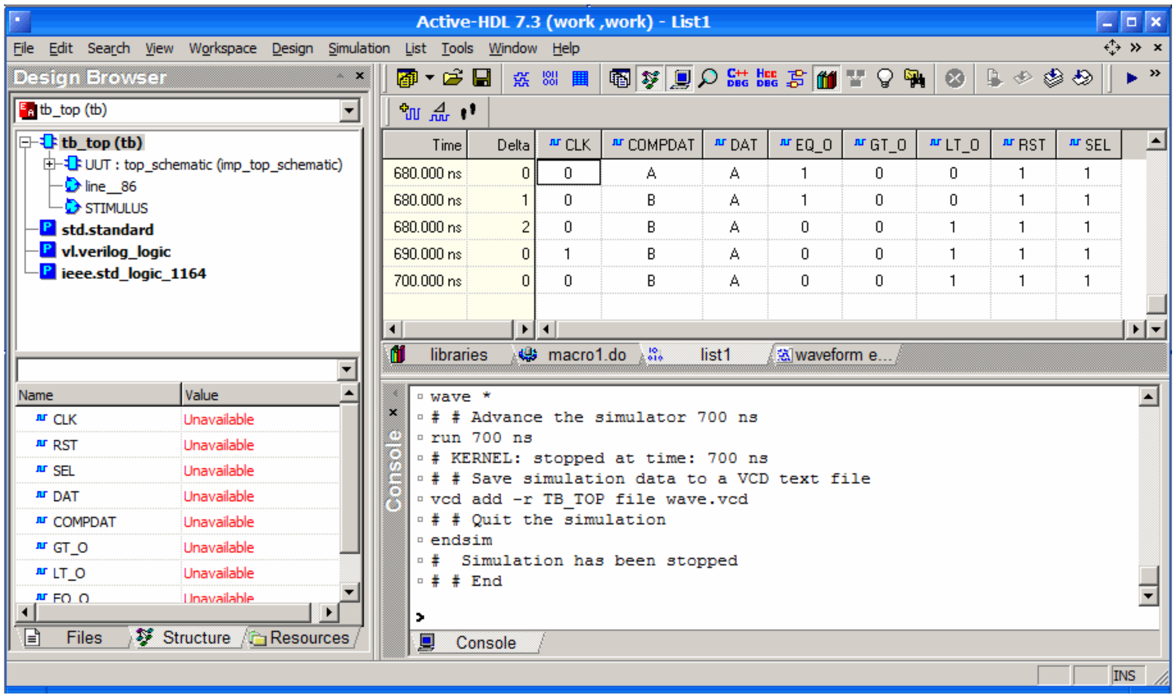
4. At the Console window command prompt, enter the following:

```
do macro1.do
```

5. Press **Enter**.

The Active-HDL macro commands are executed. Details appear in the session log. Two new windows appear with list and waveform results of the run, as shown in Figure 19.

Figure 19: Active HDL Windows Showing Run Results



Name	Value
CLK	Unavailable
RST	Unavailable
SEL	Unavailable
DAT	Unavailable
COMPDAT	Unavailable
GT_O	Unavailable
LT_O	Unavailable
EQ_O	Unavailable

Time	Delta	CLK	COMPDAT	DAT	EQ_O	GT_O	LT_O	RST	SEL
680.000 ns	0	0	A	A	1	0	0	1	1
680.000 ns	1	0	B	A	1	0	0	1	1
680.000 ns	2	0	B	A	0	0	1	1	1
690.000 ns	0	1	B	A	0	0	1	1	1
700.000 ns	0	0	B	A	0	0	1	1	1

```
wave *
# # Advance the simulator 700 ns
run 700 ns
# KERNEL: stopped at time: 700 ns
# # Save simulation data to a VCD text file
vcd add -r TB_TOP file wave.vcd
# # Quit the simulation
endsim
#  Simulation has been stopped
# # End
>
```

The Verilog value change dump (.vcd) file is stored in the .\ahdl_tutor\work\work directory. You can open this file to inspect the results or import to other software tools like the ispLEVER Power Calculator for analysis.

6. Choose **File > Exit**.

7. Choose **No** in the any Save File? dialog boxes.

   The application closes.

Task 8: Interactive Simulation

So far, the tutorial tasks have used the simulator in batch mode. In Task 3, you examined the script created by ispLEVER Project Navigator to compile source files, establish a waveform display, and run for a default period. The batch interface is ideal for rapidly verifying submodules of the design; however, to apply simulator options that will persist between sessions, you must create an Active-HDL Workspace to save environment options and a simulation file list that can be restored.

Running Active-HDL LE in Interactive Mode

In this task, you will use Active-HDL in an interactive style to manage the simulation file list and apply custom HDL compiler options.

To run Active-HDL LE in interactive mode:

1. From the ispLEVER Project Navigator, choose **Tools > Active-HDL Simulator** or click the **Active-HDL** button of the toolbar.

   Active-HDL opens, and the Getting Started dialog box appears.

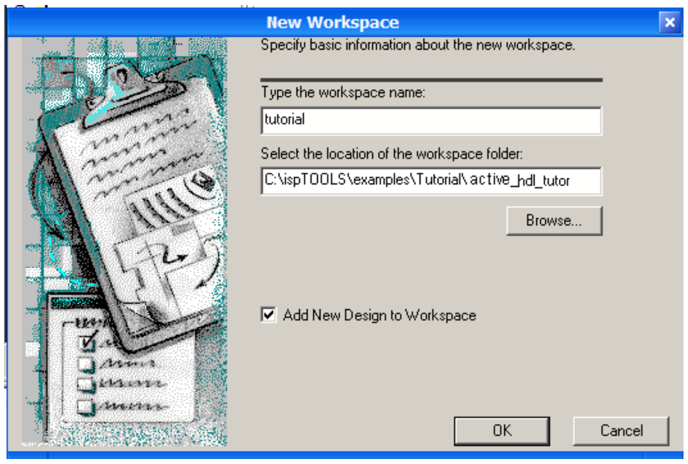
   The following steps create an Active-HDL Workspace (.aws) “container” to group designs and resource files related to the simulation. A workspace “design” refers a collection of source files, such as the RTL model and test bench of your project, the compiled library of the simulation, and output results. The Active-HDL LE license supports a single design per workspace.

2. Choose **Create new workspace** and click **OK**.

3. In the New Workspace dialog box, type the workspace name, **tutorial**, and click the **Browse** button.

4. In the Browse for Folder dialog box, select the ispTOOLS<*version*>\examples\Tutorial\active_hdl_tutor folder, as shown in Figure 20, and click **OK**.

Figure 20: Selecting the Workspace Folder in the New Workspace Dialog Box



Specify basic information about the new workspace.

Type the workspace name:

tutorial

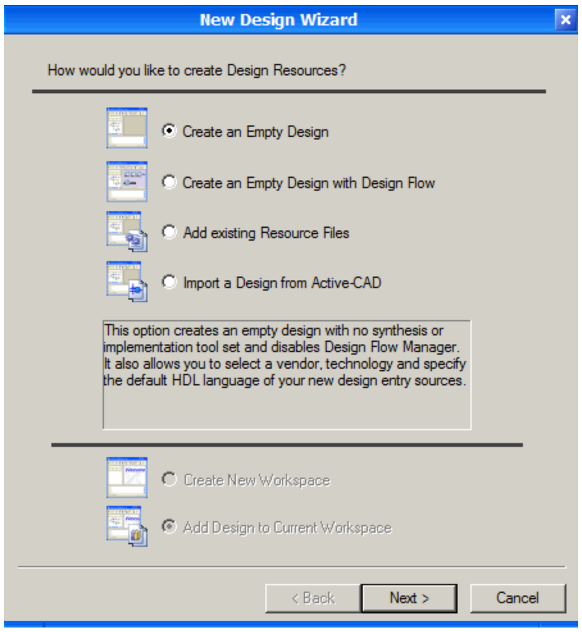
Select the location of the workspace folder:

C:\ispTOOLS\examples\Tutorial\active_hdl_tutor

☑ Add New Design to Workspace

5. In the New Design Wizard dialog box, select **Create an Empty Design**, as shown in Figure 21, and click the **Next** button.

Figure 21: Creating an Empty Design in the New Design Wizard Dialog Box



How would you like to create Design Resources?

(●) Create an Empty Design
( ) Create an Empty Design with Design Flow
( ) Add existing Resource Files
( ) Import a Design from Active-CAD

This option creates an empty design with no synthesis or implementation tool set and disables Design Flow Manager. It also allows you to select a vendor, technology and specify the default HDL language of your new design entry sources.

( ) Create New Workspace
(●) Add Design to Current Workspace

The Property Page dialog box appears.

The Create an Empty Design option configures Active-HDL LE to be used primarily for simulation tasks. The Create an Empty Design with Design Flow option enables the Design Flow Manager feature, which is an alternative to the ispLEVER Project Navigator, and provides process flow automation and controls for synthesis, simulation, and place-and-route tools. For more information on using Active-HDL as the primary “front-end” for your FPGA design work, see the Active-HDL online help and tutorials.

6. In the Property Page dialog box, select the following options:

   a. In the Block Diagram Configuration box, choose **Default HDL Language**.

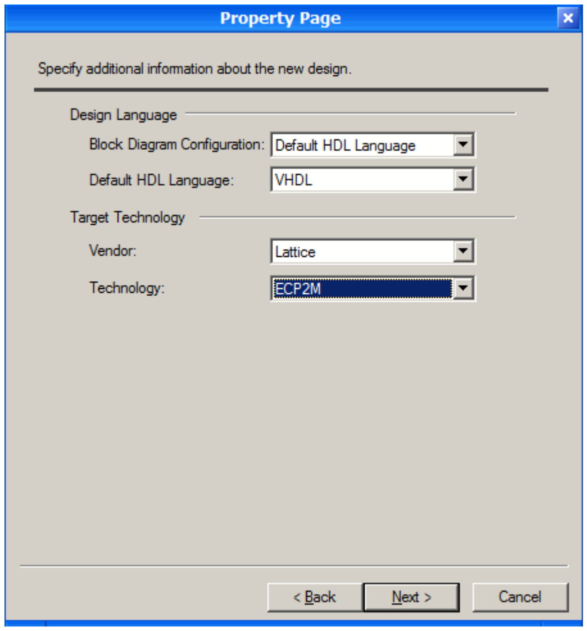
   b. In the Default HDL Language box, choose **VHDL**.

   c. In the Vendor box, choose **Lattice**.

   d. In the Technology box, choose **ECP2M**.

The Property Page dialog box should now resemble the illustration in Figure 22.

Figure 22: Settings in the Property Page Dialog Box



Specify additional information about the new design.

Design Language

Block Diagram Configuration: Default HDL Language

Default HDL Language: VHDL

Target Technology

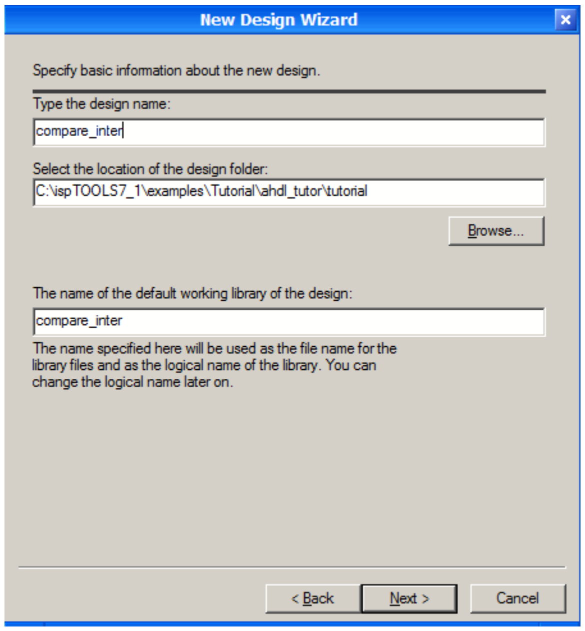
Vendor: Lattice

Technology: ECP2M

e. Click the **Next** button.

7. In the New Design Wizard dialog box, enter **compare_inter** as the design name, as shown in Figure 23.

Figure 23: Entering the Design Name in the New Design Wizard Dialog Box



Specify basic information about the new design.

Type the design name:
compare_inter

Select the location of the design folder:
C:\ispTOOLS7_1\examples\Tutorial\ahdl_tutor\tutorial

The name of the default working library of the design:
compare_inter

The name specified here will be used as the file name for the library files and as the logical name of the library. You can change the logical name later on.

8. Click the **Next** button.
9. In the final New Design Wizard dialog box, click the **Finish** button.

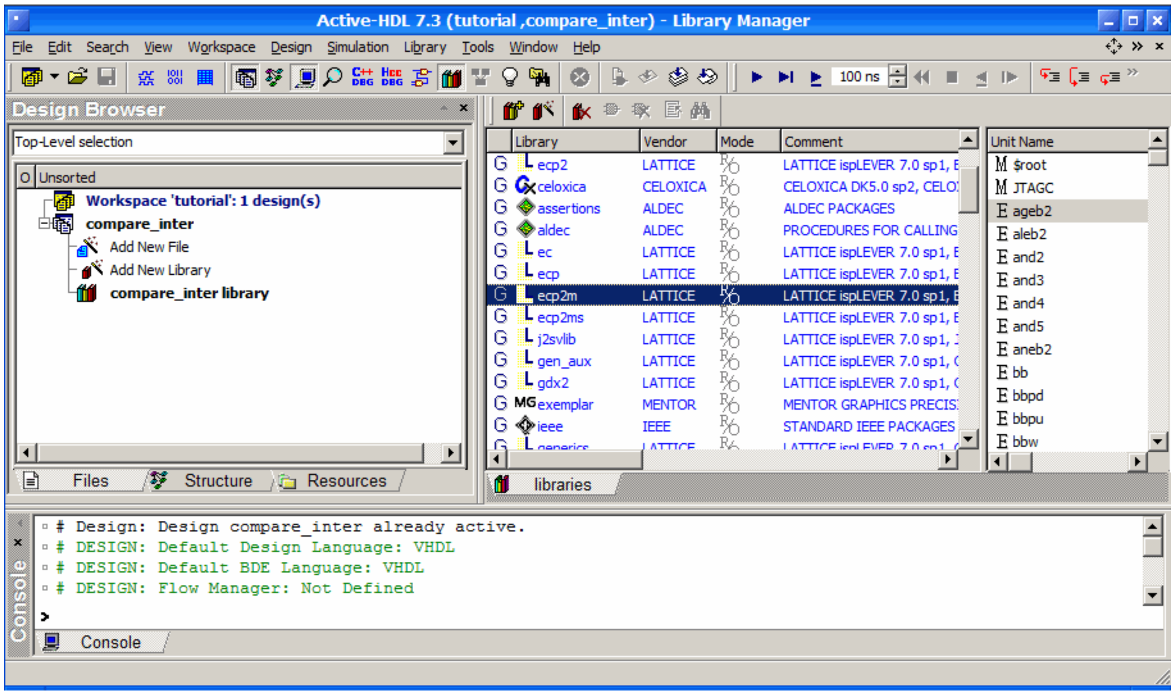
   The Design Browser appears with the new Workspace compare_inter and Library Manager tab libraries.

   The Lattice Edition license of Active-HDL provides pre-compiled Verilog and VHDL libraries of gate-level models for all Lattice Semiconductor CPLD and FPGA device families. VHDL libraries are organized by family, typically using the suffix of the device family. For example, the LatticeECP2M family appears as “ecp2” in the list. Verilog library names are preceded by “ovi_.”

   See the Active-HDL LE installation program options for details on library installation.
10. Select **ecp2m** in the Library list.

The design units compiled into ecp2 appear in the list panel to the right, as shown in Figure 24.

Figure 24: Design Units in the LatticeECP2M Family


Active-HDL 7.3 (tutorial ,compare_inter) - Library Manager
File Edit Search View Workspace Design Simulation Library Tools Window Help
100 ns
Design Browser
Top-Level selection
Unsorted
Workspace 'tutorial': 1 design(s)
compare_inter
Add New File
Add New Library
compare_inter library
Files Structure Resources
libraries
Console

	Library	Vendor	Mode	Comment
G	ecp2	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
G	celoxica	CELOXICA	R/O	CELOXICA DK5.0 sp2, CELO:
G	assertions	ALDEC	R/O	ALDEC PACKAGES
G	aldec	ALDEC	R/O	PROCEDURES FOR CALLING
G	ec	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
G	ecp	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
G	ecp2m	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
G	ecp2ms	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
G	j2svlib	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, J
G	gen_aux	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, C
G	gdx2	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, C
G	exemplar	MENTOR	R/O	MENTOR GRAPHICS PRECIS:
G	ieee	IEEE	R/O	STANDARD IEEE PACKAGES
G	generics	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1

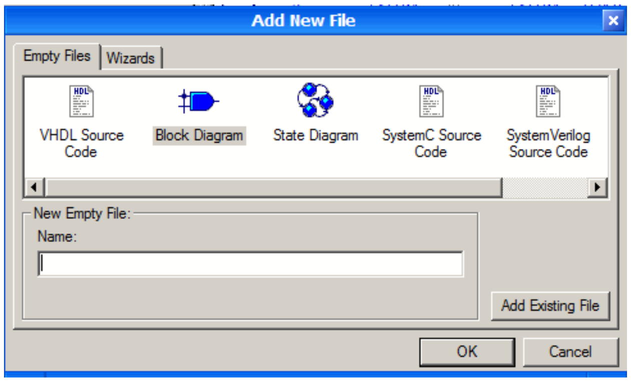
	Unit Name
M	$root
M	JTAGC
E	ageb2
E	aleb2
E	and2
E	and3
E	and4
E	and5
E	aneb2
E	bb
E	bbpd
E	bbpu
E	bbw

```
# Design: Design compare_inter already active.
# DESIGN: Default Design Language: VHDL
# DESIGN: Default BDE Language: VHDL
# DESIGN: Flow Manager: Not Defined
>
```

To add files to the design:

1. From the Design Browser, double-click on **Add New File**.
2. In the Add New File dialog box, shown in Figure 25, click the **Add Existing File** button.

Figure 25: Add New File Dialog Box



3. In the Add Files dialog box, browse to the \ispTOOLS<*version*>\examples\tutorial\active_hdl_tutor folder, and select the following source files:

- comp_eq.v
- gt.vhd
- lt.vhd
- tb_top.vhd
- top.vhd

4. Click **Add**.

   The files are listed below the compare_inter design in the Design Browser.

   The numbers in the left-hand margin of the file list indicate the order in which the files will be compiled by the simulator (1 first, 2 second, and so on.). The file list order is important to ensure that objects referenced by one model are defined before they are used. For this project, top.vhd must be in position 4, and tb_top.vhd must be in position 5.

5. Organize the file list by dragging and dropping to put top.vhd in position 4 and tb_top.vhd in position 5.

Examining HDL Compiler Options

In this task, you look at the compilation options available for Verilog and VHDL.

To examine HDL compiler options for the active design:

1. Choose **Design > Settings**.
2. From the Category list in the Design Settings dialog box, choose **Compilation > Verilog**.

   Verilog settings for the Active-HDL compiler appear. Compiler settings apply to any subsequent runs of the Verilog compiler.

3. From the Category list, choose **VHDL**.

   VHDL settings for the Active-HDL compiler appear. For example, the Standard Version option allows you to indicate against which version of the VHDL standard source the code should be compiled.

4. Disable the Enable Debug option and click **OK**.

   The Design Setting dialog box closes.

   The Enable Debug option adds information to the simulation database that supports source-level debugging features like single-stepping source code. With large gate-level designs where source-level debugging is less important, disabling this option can improve the simulator’s compilation and run-time performance.

Note

Options in the Design Setting dialog boxes apply to the current workspace or design only. If you want to apply default settings for future workspaces, see the Tools > Preferences dialog boxes.

Compiling Design Source Files

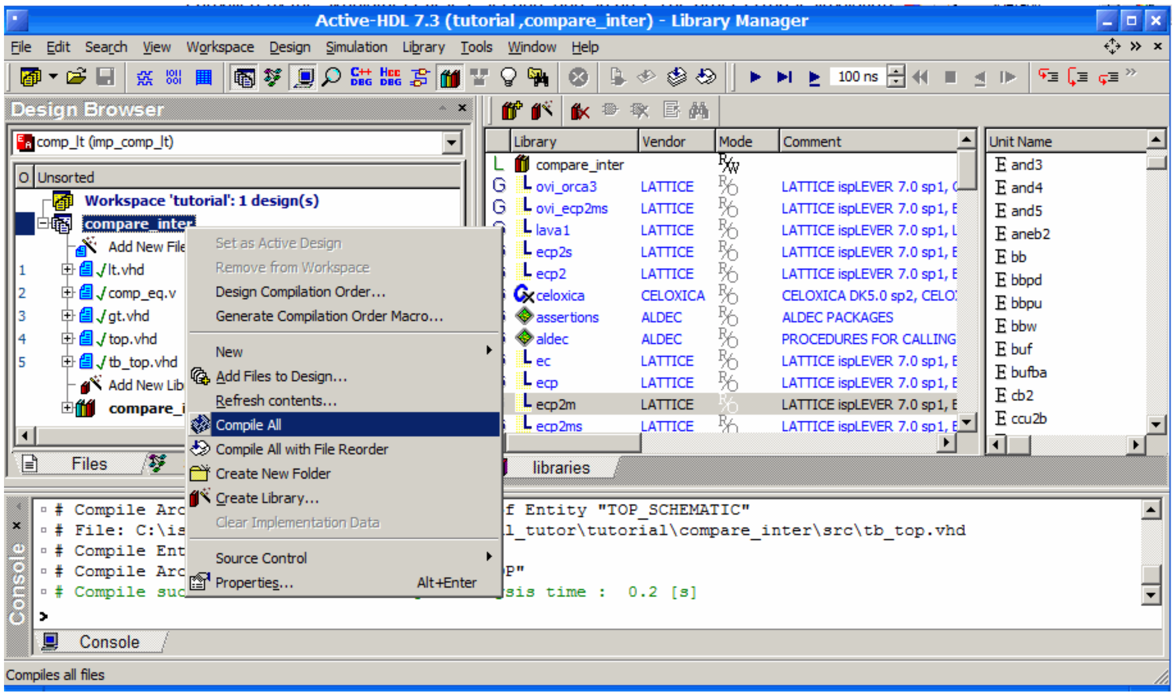
Now you will compile the source files.

To compile design source files:

1. Choose the **compare_inter** design from the Design Browser.
2. Right-click and choose **Compile All**, as shown in Figure 26.

   The simulator performs a syntax check, then compiles the source files into the compare_inter library.

Figure 26: Compiling the Source Files



	Library	Vendor	Mode	Comment
L	compare_inter		R/W	
G	ovi_orca3	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, (
G	ovi_ecp2ms	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
	lava1	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, L
	ecp2s	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
	ecp2	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
	celoxica	CELOXICA	R/O	CELOXICA DK5.0 sp2, CELO:
	assertions	ALDEC	R/O	ALDEC PACKAGES
	aldec	ALDEC	R/O	PROCEDURES FOR CALLING
	ec	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
	ecp	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
	ecp2m	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E
	ecp2ms	LATTICE	R/O	LATTICE ispLEVER 7.0 sp1, E

Unit Name
E and3
E and4
E and5
E aneb2
E bb
E bbpd
E bbpu
E bbw
E buf
E bufba
E cb2
E ccu2b

```
# Compile Arc                              f Entity "TOP_SCHEMATIC"
# File: C:\is                              l_tutor\tutorial\compare_inter\src\tb_top.vhd
# Compile Ent
# Compile Arc                              P"
# Compile suc                              sis time :  0.2 [s]
>
```

Information about the compilation appears in the session log of the Console window.

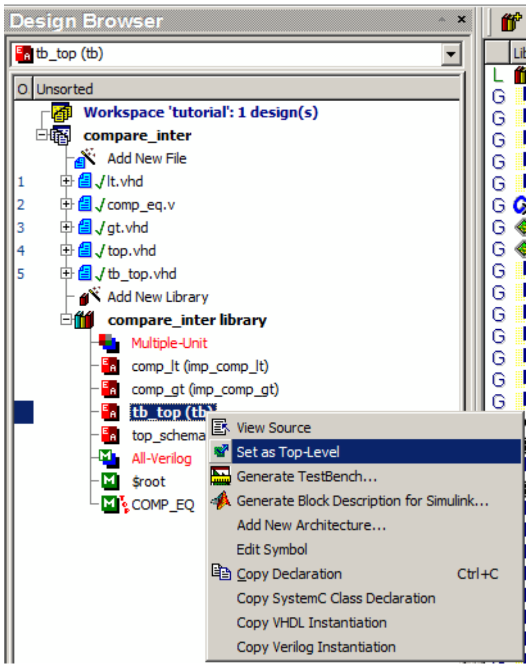
3. From the Design Browser, expand the **compare_inter** library.

   All modules of the project appear in the list. Icons are used to differentiate between VHDL and Verilog objects.

4. Select **tb_top (tb)** from the list, right-click, and choose **Set as Top-Level**, as shown in Figure 27.

The library entry appears in bold font.

Figure 27: Specifying the Top-Level Library



5. From the Design Browser, select the **Structure** tab.

   The design hierarchy appears with tb_top as the root of the simulation.

6. Choose **File > New > Waveform**.

   A waveform editor appears.

7. Choose **Simulation > Initialize**.

   The simulator engine elaborates the database and initializes all signal and variable states of the model on the basis of the top-level selection made earlier.

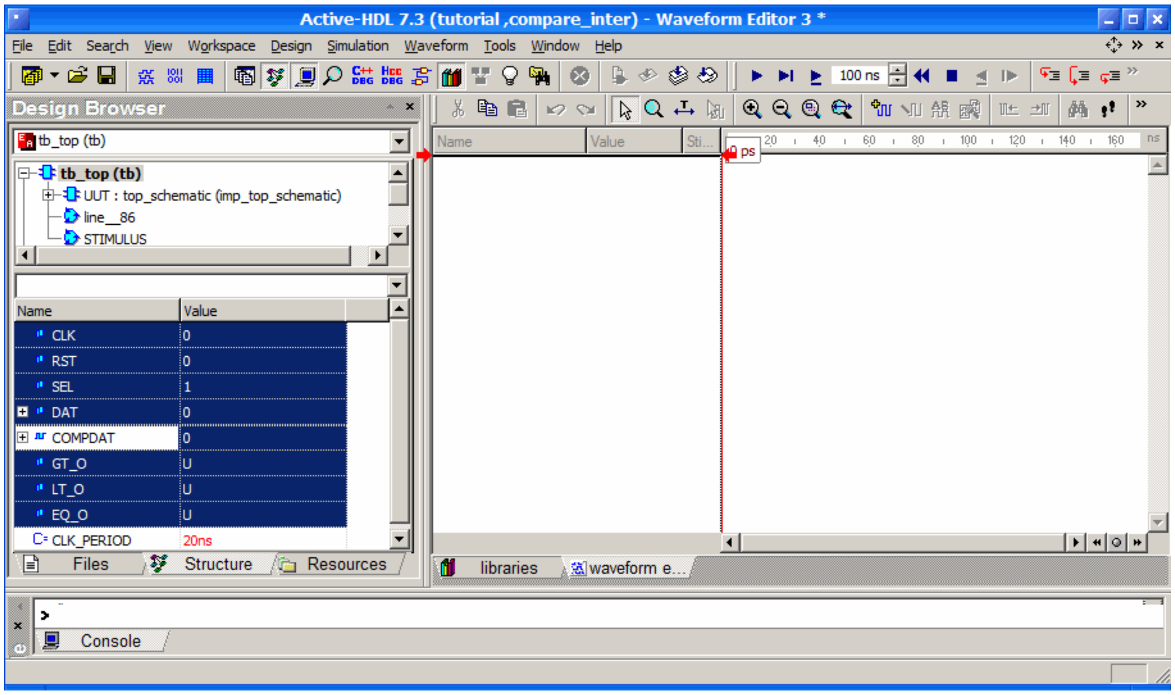
8. From the Design View, select **tb_top (tb)**.

   Signals and constants of the test bench appear in the list.

9. Select all the signals, then drag and drop them to the Waveform Editor.

The signals appear in the Name list of the Waveform Editor, as shown in Figure 28.

Figure 28: Selecting the Test Bench Signals


Active-HDL 7.3 (tutorial ,compare_inter) - Waveform Editor 3 *
File Edit Search View Workspace Design Simulation Waveform Tools Window Help
100 ns
Design Browser
tb_top (tb)
Name Value Sti...
ps
20 40 60 80 100 120 140 160 ns
Files Structure Resources
libraries waveform e...
Console

```
tb_top (tb)
  UUT : top_schematic (imp_top_schematic)
  line__86
  STIMULUS
```

Name	Value
CLK	0
RST	0
SEL	1
DAT	0
COMPDAT	0
GT_O	U
LT_O	U
EQ_O	U
CLK_PERIOD	20ns

10. Choose **Simulation > Run Until**.
11. In the Run Until dialog box, enter **700 ns** and click **OK**.

    The simulator advances to time 700 ns.
12. Click the **Zoom to Fit** button of the Waveform Editor toolbar.
13. Choose **Simulation > End Simulation**.

    The simulator is initialized and clears memory.

    You can now choose to simulate with a new top level or adjust the project to add more files.
14. Choose **File > Exit**.

    Active-HDL LE closes.

Summary

You have completed the Active-HDL Lattice Edition Tutorial. In this tutorial, you have learned how to do the following:

- Set up Project Navigator to launch Active-HDL LE as the default simulator.
- Examine interface scripts issued by Project Navigator.

- Browse simulation objects like signals, ports, component instances, and processes.
- Set source and signal state breakpoints to isolate behavior of the model.
- Create custom waveform displays.
- Measure the time between waveform events.
- Create a macro script to automate several simulator commands.
- Set up an Active-HDL workspace to save the simulation file list and compiler options.

Glossary

Following are the terms and concepts that you should understand to use this tutorial effectively.

Active-HDL LE Active-HDL LE is an abbreviation of Active-HDL Lattice Edition.

HDL An HDL is a hardware description language, which describes the structure and function of integrated circuits.

simulation Simulation is a technology to help verify the behavior of an FPGA design to confirm functional correctness and performance.

synthesis Synthesis is the process of translating a high-level design (RTL) description consisting of state machines, truth tables, Boolean equations, or all three into a process-specific, gate-level logic implementation.

Verilog Verilog is a language for describing the structure and function of integrated circuits.

VHDL VHDL (or VHSIC (Very High-Speed Integrated Circuits) Hardware Description Language) is a language for describing the structure and function of integrated circuits.

Recommended References

You can find additional information on the subjects covered by this tutorial from the following recommended sources:

- Active-HDL Lattice Edition online documentation and tutorials
- support.aldec.com
- ispLEVER online documentation