



Schematic and ABEL-HDL Design Tutorial

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

October 2005

Copyright

Copyright © 2005 Lattice Semiconductor Corporation.

This document may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Lattice Semiconductor Corporation.

Trademarks

Lattice Semiconductor Corporation, L Lattice Semiconductor Corporation (logo), L (stylized), L (design), Lattice (design), LSC, E2CMOS, GAL, GDX, Generic Array Logic, ISP, ispATE, ispCLOCK, ispDOWNLOAD, ispGAL, ispGDS, ispGDX, ispGDXV, ispGDX2, ispGENERATOR, ispJTAG, ispLEVER, ispLEVERCORE, ispLSI, ispMACH, ispPAC, ispTRACY, ispTURBO, ispVIRTUAL MACHINE, ispVM, ispXP, ispXPGA, ispXPLD, LatticeEC, LatticeECP, LatticeECP-DSP, LatticeXP, MACH, MachXO, ORCA, PAC, PAC-Designer, PAL, Performance Analyst, Silicon Forest, Speedlocked, Speed Locking, SuperBIG, SuperCOOL, SuperFAST, SuperWIDE, sysCLOCK, sysCONFIG, sysDSP, sysHSI, sysI/O, sysMEM, The Simple Machine for Complex Design, TransFR, UltraMOS, XPIO, and specific product designations are either registered trademarks or trademarks of Lattice Semiconductor Corporation or its subsidiaries in the United States and/or other countries. ISP and Bringing the Best Together are service marks of Lattice Semiconductor Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Disclaimers

NO WARRANTIES: THE INFORMATION PROVIDED IN THIS DOCUMENT IS "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF ACCURACY, COMPLETENESS, MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL LATTICE SEMICONDUCTOR CORPORATION (LSC) OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (WHETHER DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION PROVIDED ON THIS SITE, EVEN IF LSC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF CERTAIN LIABILITY, SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

LSC may make changes to these materials, specifications, or information, or to the products described herein, at any time without notice. LSC makes no commitment to update this documentation. LSC reserves the right to discontinue any product or service without notice and assumes no obligation 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.

Contents

Schematic and ABEL-HDL Design Tutorial	1
Learning Objectives	1
Time to Complete This Tutorial	2
System Requirements	2
Accessing Online Help	2
About the Tutorial Design	2
About the Tutorial Data Flow	4
Module 1: Design Entry and Simulation	5
Learning Objectives	5
Time to Complete This Module	5
Lesson 1: Setting Up a Project	5
Lesson 2: Creating a Top-Level Schematic Source	10
Lesson 3: Finishing the Schematic	32
Lesson 4: Adding ABEL-HDL Sources to the Project	36
Lesson 5: Navigating the Design	40
Lesson 6: Running Functional Simulation	46
Module 2: Design Implementation	51
Prerequisites	51
Learning Objectives	51
Time to Complete This Module	51
Task 1: Pre-Assign Pin Locations	51
Task 2: Fit the Design	57
Task 3: Set Report Viewing Options	58
Task 4: Read the Fitter Report	59
Module 3: Design Verification	61
Prerequisites	61
Learning Objectives	61

Time to Complete This Module	62
Task 1: Perform Static Timing Analysis	62
Task 2: Run Timing Simulation	64
Task 3: Analyze the Simulation Results	66
Task 4: Correlate Simulation Results by Cross-Probing	69
Task 5: Timing Simulation for Third-Party Simulators	71
Task 6: Board-Level Static Timing Analysis	73
Summary	74
Glossary	75
Recommended Reference Materials	76

Schematic and ABEL-HDL Design Tutorial

This tutorial leads you through all the basic steps of designing, simulating, implementing, and verifying a counter circuit targeted to a CPLD device. It shows you how to use several processes, tools, and reports from the ispLEVER software suite to create a top-level schematic and add new ABEL-HDL sources to the project. ABEL-HDL is a hierarchical hardware description language that supports a variety of behavioral input forms, including high-level equations, state diagrams, and truth tables. The tutorial then proceeds to step through the process of performing functional simulation, applying constraints, fitting the design, performing timing simulation, and analyzing the results.

The ispLEVER software supports mixed-mode design entry: a design with at least one schematic module as the top project source, and one or more sources of the same language. The language sources are mutually exclusive, so you must choose one of the three types when you begin a new project. For example, in addition to the schematic, you must choose an ABEL-HDL source, a Verilog HDL source, or a VHDL source.

Learning Objectives

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

- ◆ Set up an ispLEVER schematic project
- ◆ Create a top-level schematic source
- ◆ Create and import ABEL-HDL sources into the project
- ◆ Navigate the design
- ◆ Import a test vector file and run functional simulation

- ◆ Set pin constraints, fit the design, and read the fitter report
- ◆ Run timing analysis and simulation and analyze the results
- ◆ Correlate simulation results using cross-probing
- ◆ Export the netlist and delays for timing simulation
- ◆ Build board-level stamp models of a design

Time to Complete This Tutorial

The time to complete this tutorial is about 90 minutes.

System Requirements

One of the following software configurations is required to complete the tutorial:

- ◆ ispLEVER Base
- ◆ ispLEVER Advanced
- ◆ IspLEVER Advanced Systems

Accessing Online Help

You can find online help information on any tool included in the tutorial at any time by pressing the F1 key.

About the Tutorial Design

The design in this tutorial consists of a top-level schematic and two lower-level ABEL-HDL modules. It uses a top-down, bottom-up design methodology. The design that you create and simulate is targeted to an ispMACH 5000VG CPLD device.

This tutorial is divided into three modules, with Module 1 divided into six lessons. You should start with Module 1 and continue to the end of Module 3. The modules cover the following topics:

- ◆ Module 1: Design Entry and Simulation

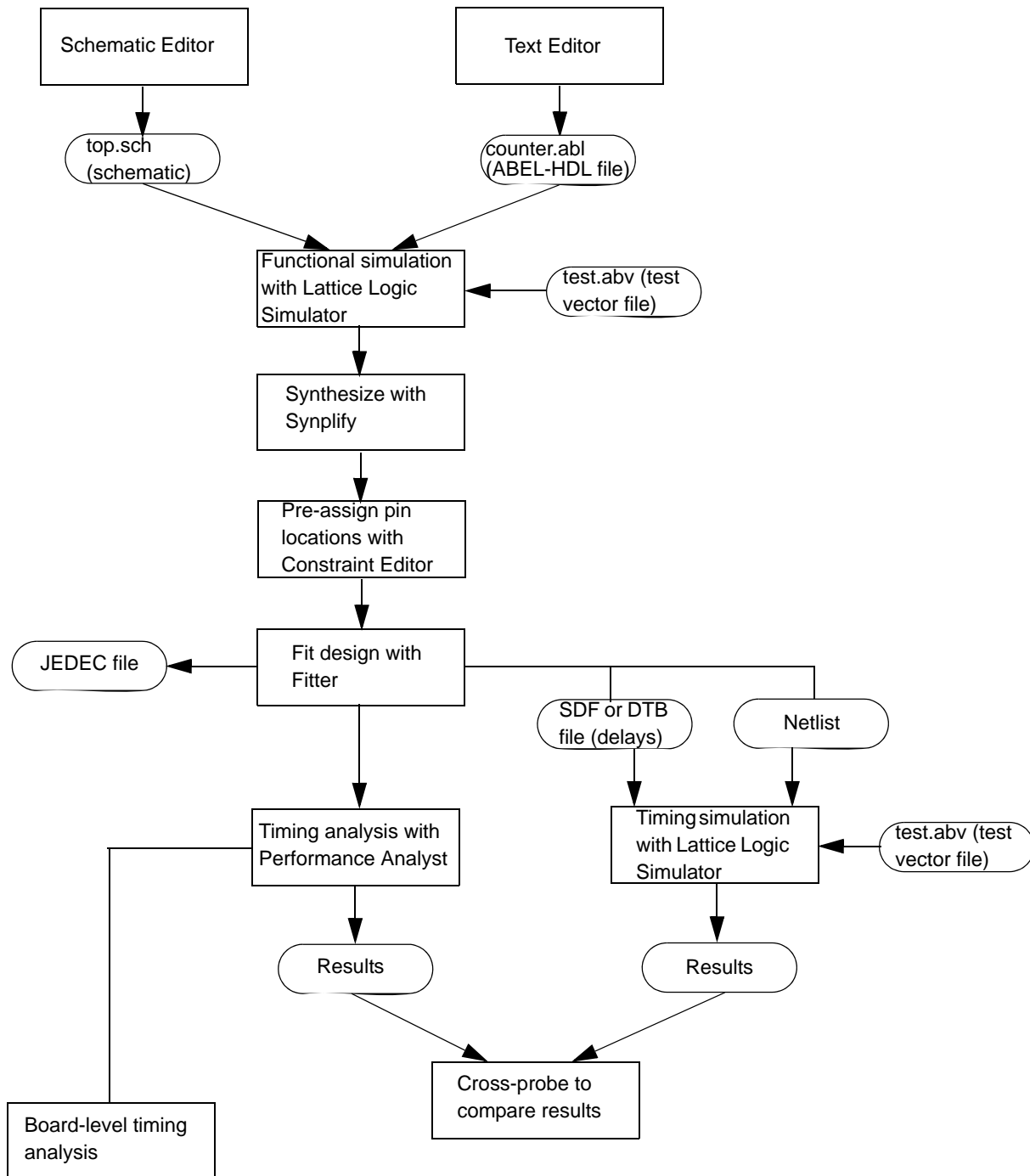
This module shows you how to create a top-level schematic, then how to add two other source ABEL-HDL modules to the project. You learn how to navigate through the schematic. At the end of the module, you import a test vector file and perform functional simulation to detect any logic errors.
- ◆ Module 2: Design Implementation

This module shows you how to set pin constraints, fit the design, and read the fitter report.
- ◆ Module 3: Design Verification

This module shows you how to use the Performance Analyst to perform static timing analysis, the Lattice Logic Simulator to perform timing simulation, and the Waveform Viewer to analyze the simulation results. You also learn how to use the Waveform Viewer to display the simulation results on the schematics. Finally, the tutorial shows you how to generate SDF and DTB files containing the netlist and delays to use in third-party timing simulators and stamp model files to use with third-party board-level static timing analysis tools.

About the Tutorial Data Flow

The following figure illustrates the tutorial data flow through the system. You may find it helpful to refer to this diagram as you move through the tutorial tasks.



Module 1: Design Entry and Simulation

In this module, you will set up an ispLEVER project and create a top-level schematic, then add two other modules to the project. You will learn how to use the basic functions of the Hierarchy Navigator to traverse the design. Finally, you will import a test vector file and perform functional simulation to detect any logic errors.

The schematic design entry environment is a set of tools that allow you to capture the structure of a design as either a flat description or a hierarchical set of components, and the connectivity between these components. Then you can use this description to drive the fitter and verification tools. Designs can be single-level (flat) or multi-level (hierarchical). Schematics can be drawn on multiple “sheets” and be any size.

Learning Objectives

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

- ◆ Add various schematic elements to create a top-level schematic source
- ◆ Check the schematic for errors
- ◆ Use a text editor to create a new ABEL-HDL source
- ◆ Import an ABEL-HDL source into the project
- ◆ Use the Hierarchy Navigator to navigate through the design and to try “debug” methods
- ◆ Use the Functional Simulator process to verify the behavior of your design

Time to Complete This Module

The time to complete this module is about 50 minutes.

Lesson 1: Setting Up a Project

In Lesson 1, you will set up the tutorial 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.

The following tasks are covered in this lesson:

- ◆ Task 1: Create a New Project
- ◆ Task 2: Target a Device
- ◆ Task 3: Copy Schematic Symbols to the Project Directory

Task 1: Create a New Project

To begin a new project, you need to create a project directory. Then you must give the project file a name with an .syn suffix. The Project Navigator will use this file name later to reload the project.

To create a new project:

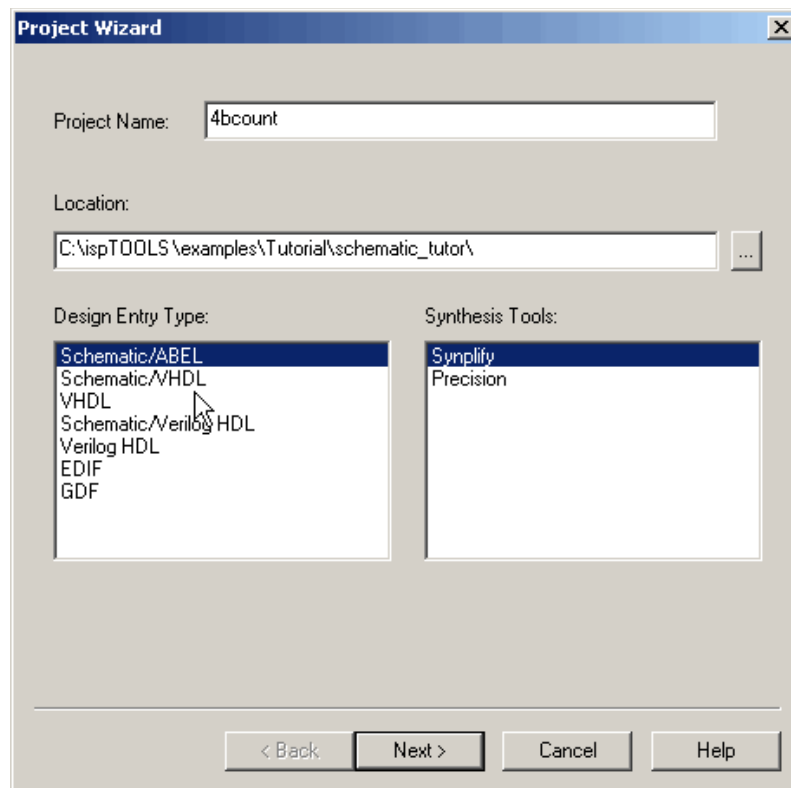
1. Start the ispLEVER system, if it is not already running.
2. In the Project Navigator, choose **File > New Project** to open the Project Wizard dialog box.
3. In this dialog box, do the following:
 - a. In the Project name box, type **4bcount**.
 - b. In the Location box, change to the following directory:

`<install_path>\examples\Tutorial\schematic_tutor`

Note:

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

- c. In the Design Entry Type box, select **Schematic/ABEL**.
- d. In the Synthesis Tools box, select **Synplify**.
- e. Click **Next**.



The Project Wizard – Select Device dialog box appears.

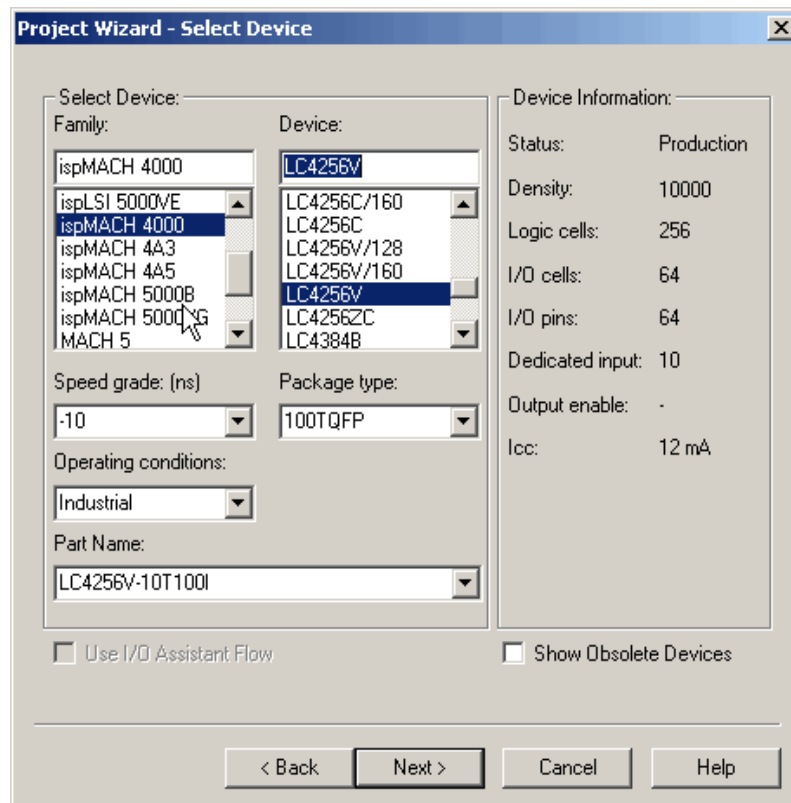
Task 2: Target a Device

In the Project Navigator Sources window is the device icon  next to the target device for the project. The Project Navigator enables you to target a

design to a specific Lattice device at any time during the design process. The default device is the ispLSI526VE-165LF256. For this project, you will target a different device.

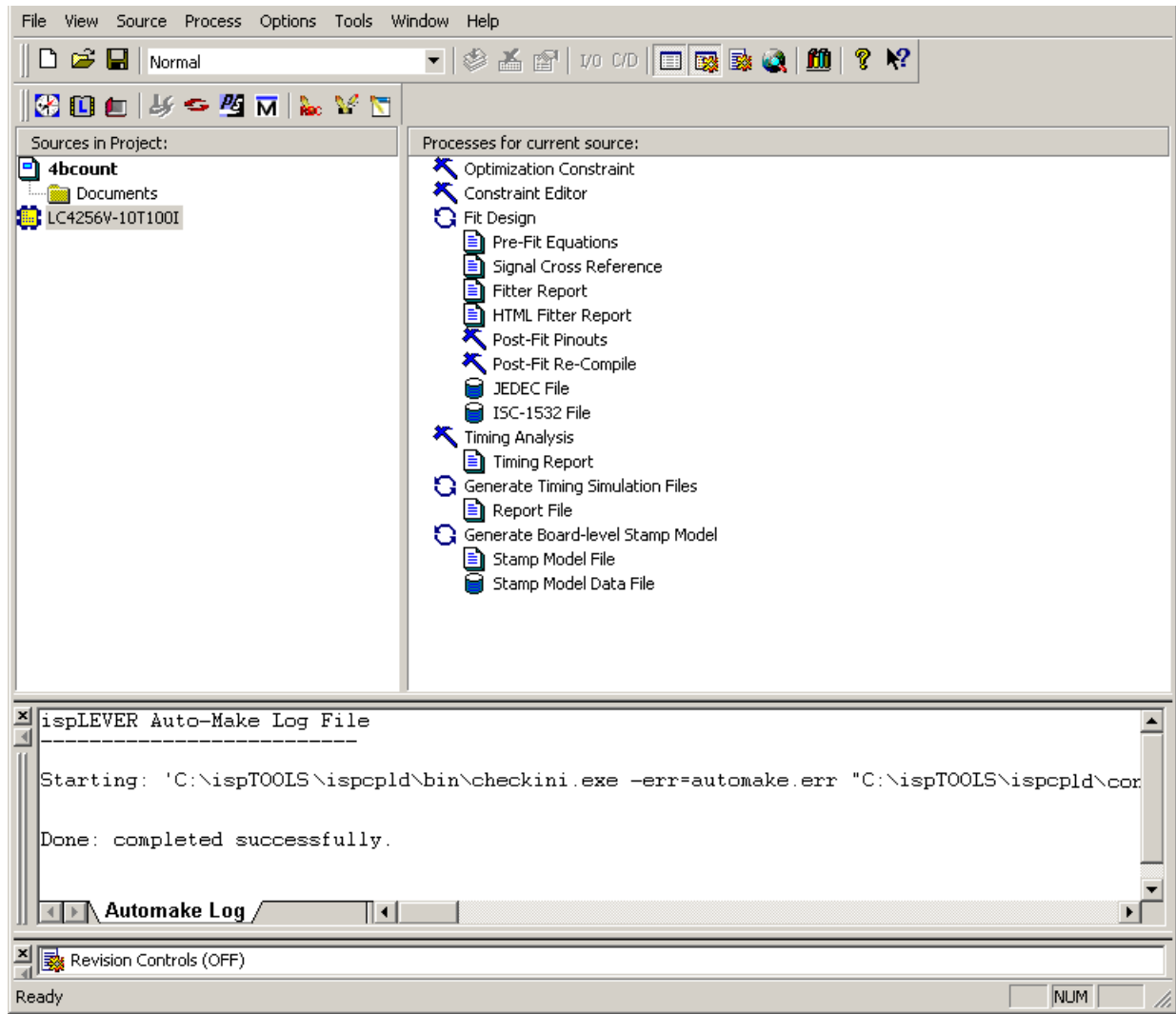
To view the list of available devices and to change the target device:

1. In the Project Wizard – Select Device dialog box, do the following:
 - a. In the Family box, choose **ispMACH 4000** from the drop-down list.
 - b. In the Device box, choose **LC4256V**.
 - c. Accept the default settings and click **Next**.



2. Click **Next** to activate the Add Source dialog box.
3. Click **Next** in the Add Source dialog box, then click **Finish**.

Your Project Navigator should look like the following illustration.



Task 3: Copy Schematic Symbols to the Project Directory

A schematic is composed of symbols, wires, I/O markers, graphics, and text. Symbols are graphic representations of components. The term “symbol” usually refers to an electrical symbol, such as a gate or a subcircuit. You can draw graphic-only symbols, such as title blocks, with the Symbol Editor, but these have no electrical meaning.

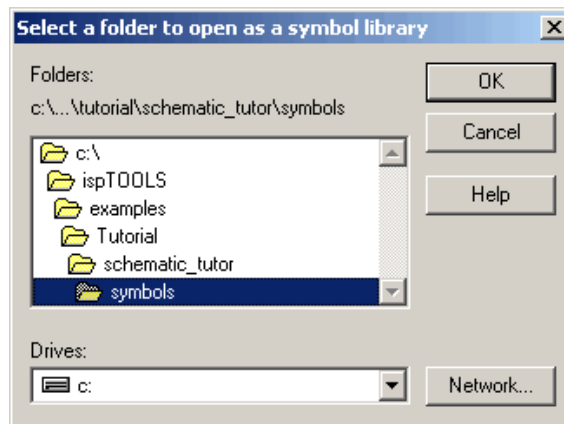
Symbols are the most basic elements of a schematic. Symbols represent primitive design elements, whether those elements are individual transistors, complete gates, or a complex integrated circuit. A symbol can also be the hierarchical representation of a subcircuit (a “Block” symbol).

In this task, you will copy two symbol files to your project directory so that you can use these pre-made symbols in your design.

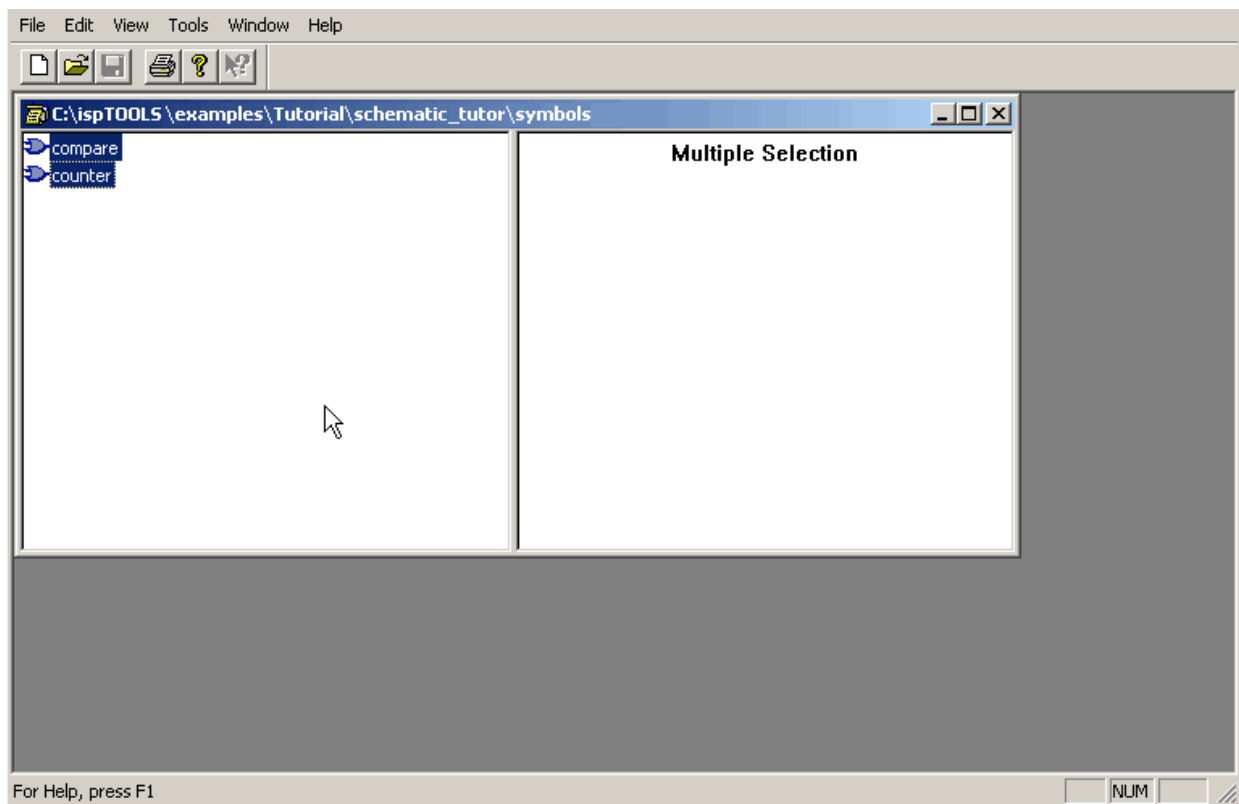
To copy schematic symbols to the project directory:

1. In the Project Navigator, choose **Window > Library Manager**.
2. In the Library Manager, choose **File > Open Folder** to open the Select a Folder to Open as a Symbol Library dialog box.
3. Change to the following directory and click **OK**:

`<install_path>\examples\Tutorial\schematic_tutor\symbols`

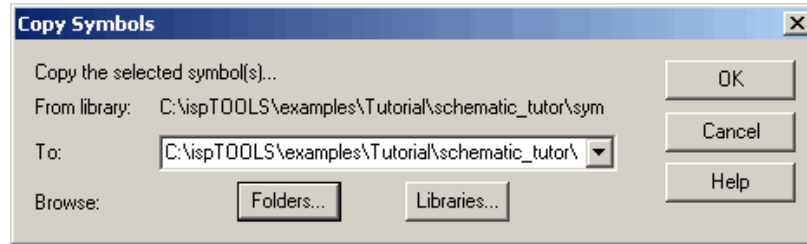


4. Select the two symbols (**compare** and **counter**), using Shift-click to select the second symbol..



5. Choose **Edit > Copy Symbol(s)** to open the Copy Symbols dialog box.

- Click **Folders**, then change to the `\schematic_tutor` directory and click **OK**.



- Click **OK** to close the Copy Symbols dialog box.
- Choose **File > Exit** to close the Library Manager.

Lesson 2: Creating a Top-Level Schematic Source

The schematic design entry environment is a set of tools that enable you to capture the structure of a design as either a flat description or a hierarchical set of components, and the connectivity between these components. Then you can use this description to drive the fitter and verification tools.

You create schematic designs using the Schematic Editor. Schematics can be single-level (flat) or multi-level (hierarchical). Schematics can be drawn on multiple “sheets” and be any size. The Schematic Editor can work in conjunction with the Hierarchy Navigator, Symbol Editor, and Library Manager programs.

In this lesson, you will begin creating a top-level schematic source for the project.

Note

If you want to skip this lesson on creating a schematic, you can go directly to Lesson 3.

The following tasks are covered in this lesson:

- ◆ Task 4: Add a New Schematic to the Project
- ◆ Task 5: Resize the Schematic Sheet
- ◆ Task 6: Place Two Block Symbols from the Local Symbol Library
- ◆ Task 7: Place a Symbol from the REGS Generic Symbol Library
- ◆ Task 8: Place Symbols from the IOPAD Generic Symbol Library
- ◆ Task 9: Add Wires and Buses
- ◆ Task 10: Edit the Schematic
- ◆ Task 11: Add Wires to Connect the Symbols
- ◆ Task 12: Duplicate the Input Pad and Wire Stub
- ◆ Task 13: Name the Buses
- ◆ Task 14: Add Bus Taps with Signal Names

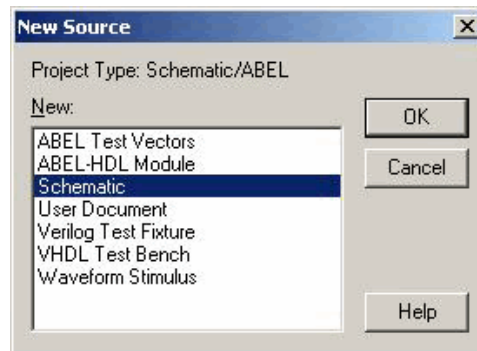
- ◆ Task 15: Add Input Net Names
- ◆ Task 16: Add Data Input Net Names
- ◆ Task 17: Create Iterated Instances of the Flip-Flop
- ◆ Task 18: Add Input Markers

Task 4: Add a New Schematic to the Project

Designing top-down, you will create the top-level source for the project. Because this design is mixed schematic and ABEL-HDL, you can use either as a source at the top-level design file. However, for this tutorial you will use a schematic.

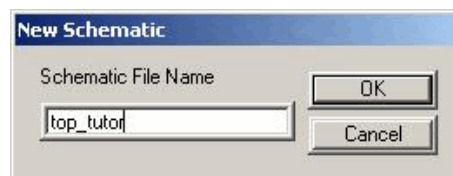
To add a new schematic source to the project:

1. In the Project Navigator, choose **Source > New** to open the New Source dialog box.
2. Select **Schematic** and click **OK**.



The Schematic Editor opens and prompts you to enter a file name for the schematic.

3. Type the name **top_tutor** and click **OK**.



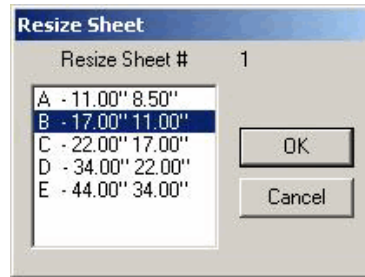
The Schematic Editor names the current schematic sheet `top_tutor`, and the software imports the schematic into the Project Navigator as a new source file, `top_tutor.sch`.

Task 5: Resize the Schematic Sheet

You can resize a schematic sheet using the Resize command. The Resize command takes effect immediately and is applied to the sheet selected in the Sheets dialog box, not the active sheet or the sheet currently being worked on.

To resize the schematic sheet:

1. In the Schematic Editor, choose **File > Sheets** to open the Sheets dialog box. Because there is only one sheet in the schematic, you cannot select another sheet.
2. Click **Resize** to open the Resize Sheet dialog box. The current size of the selected sheet is highlighted. Other available sheet size choices are listed.
3. Select **B**, and then click **OK** to close the Resize Sheet dialog box.



4. In the Sheets dialog box, click **Open**.

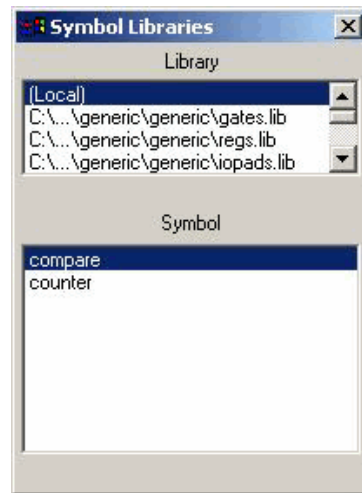
The software resizes the sheet in the Schematic Editor.

Task 6: Place Two Block Symbols from the Local Symbol Library

The first step in this top-down design is to create block symbols to represent lower-level modules in the design. Later, you will design the I/O ports of the lower-level ABEL-HDL sources to match the names of the pins on the corresponding block symbols.

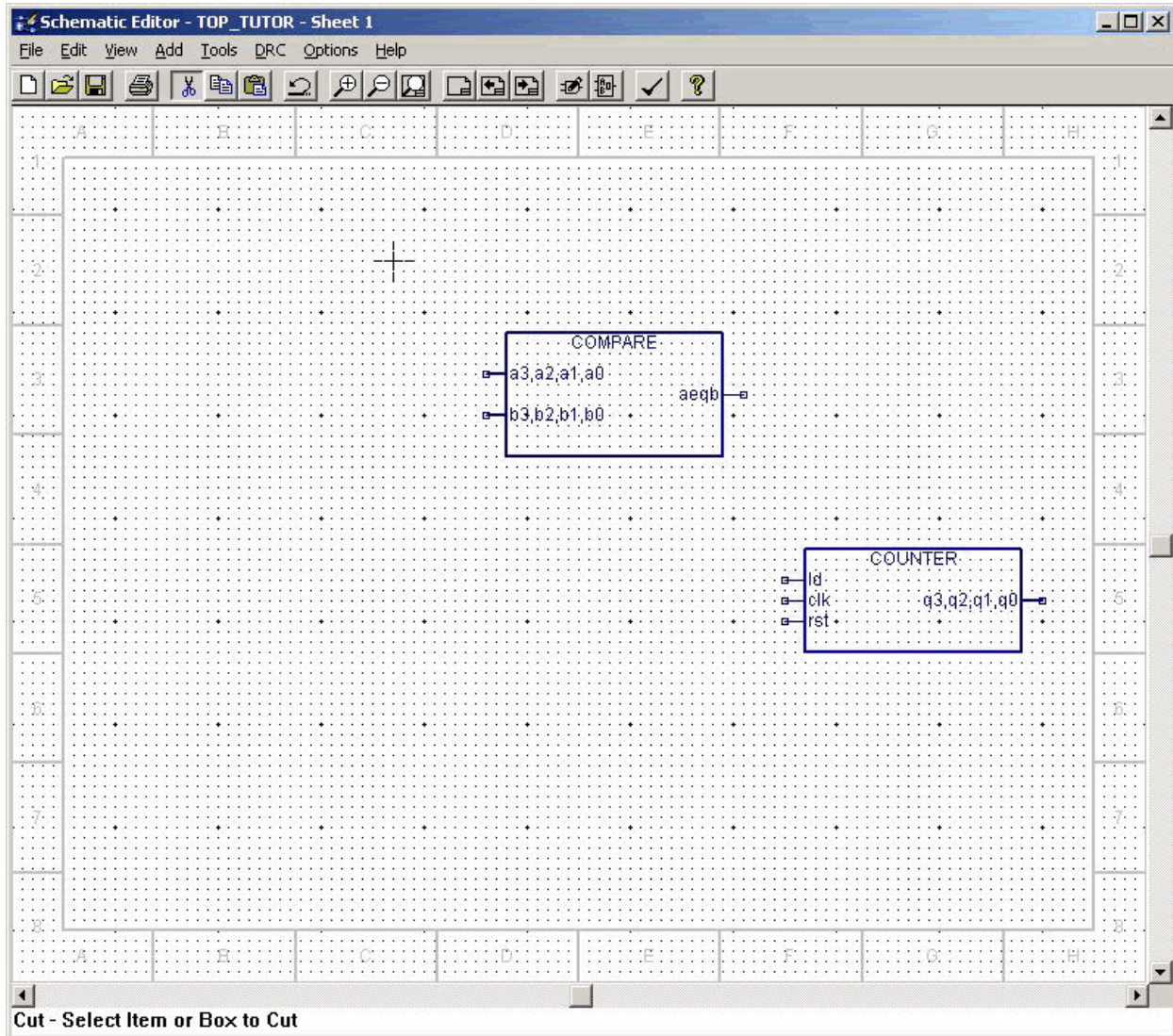
To place the block symbols in the schematic:

1. In the Schematic Editor, choose **Add > Symbol** to open the Symbol Libraries dialog box.
2. In the dialog box, under Library, select **Local**. This library is in the project folder you created. You copied two block symbols into it at the beginning of this tutorial in Task 3.



3. Under Symbol, select **compare**. The symbol is attached to the cursor. (If you do not see a grid, choose **Options > Preferences** and select **Display Grid**.)
4. Position the cursor slightly above the upper middle area of the schematic, and then click to place the symbol.
5. Right-click to remove the symbol from the cursor. (If you make a mistake placing a symbol, you can choose **Edit > Cut** or **Move**, and then click the symbol that is to receive the action.)
6. Now select the **counter** symbol.

7. Position the cursor below and to the right of the COMPARE symbol, and then click to place the symbol.



8. Again, right-click to remove the symbol from the cursor.
9. Leave the dialog box open. You will continue to use it in the next step.

Note

It is not critical that your schematic look exactly like the example. Yours may look different, depending on unimportant factors that do not affect the proper use of this tutorial.

Task 7: Place a Symbol from the REGS Generic Symbol Library

The target device determines which symbol libraries are available. If you use symbols from the Generic Symbol Library, you can migrate designs to different devices without having to redraw the schematic.

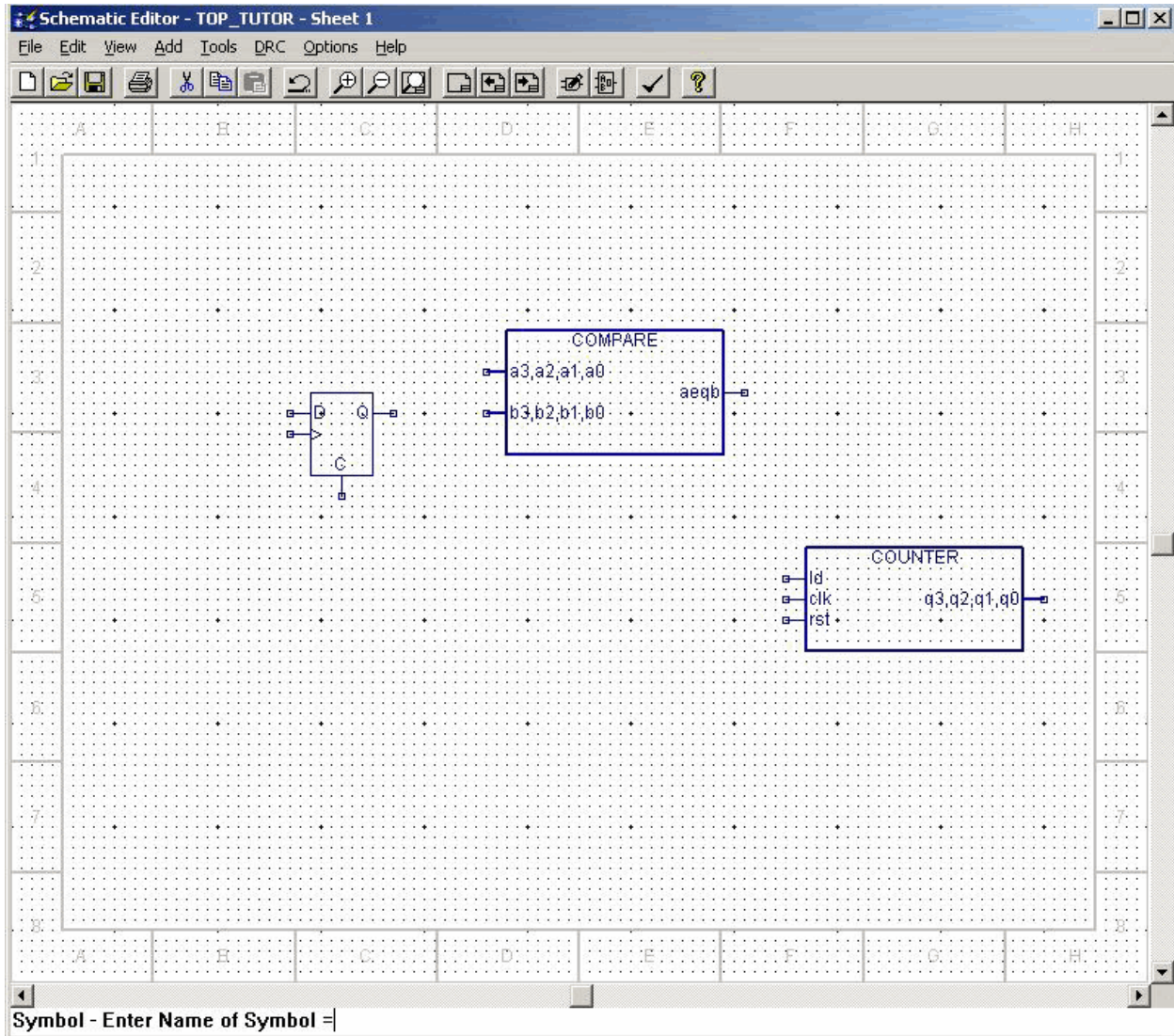
Note:

You can use the Drawing Toolbar to add symbols and other schematic drawing functions. To display the Drawing Toolbar, choose **View > Drawing Toolbar**.

To place a symbol from the REGS generic symbol library:

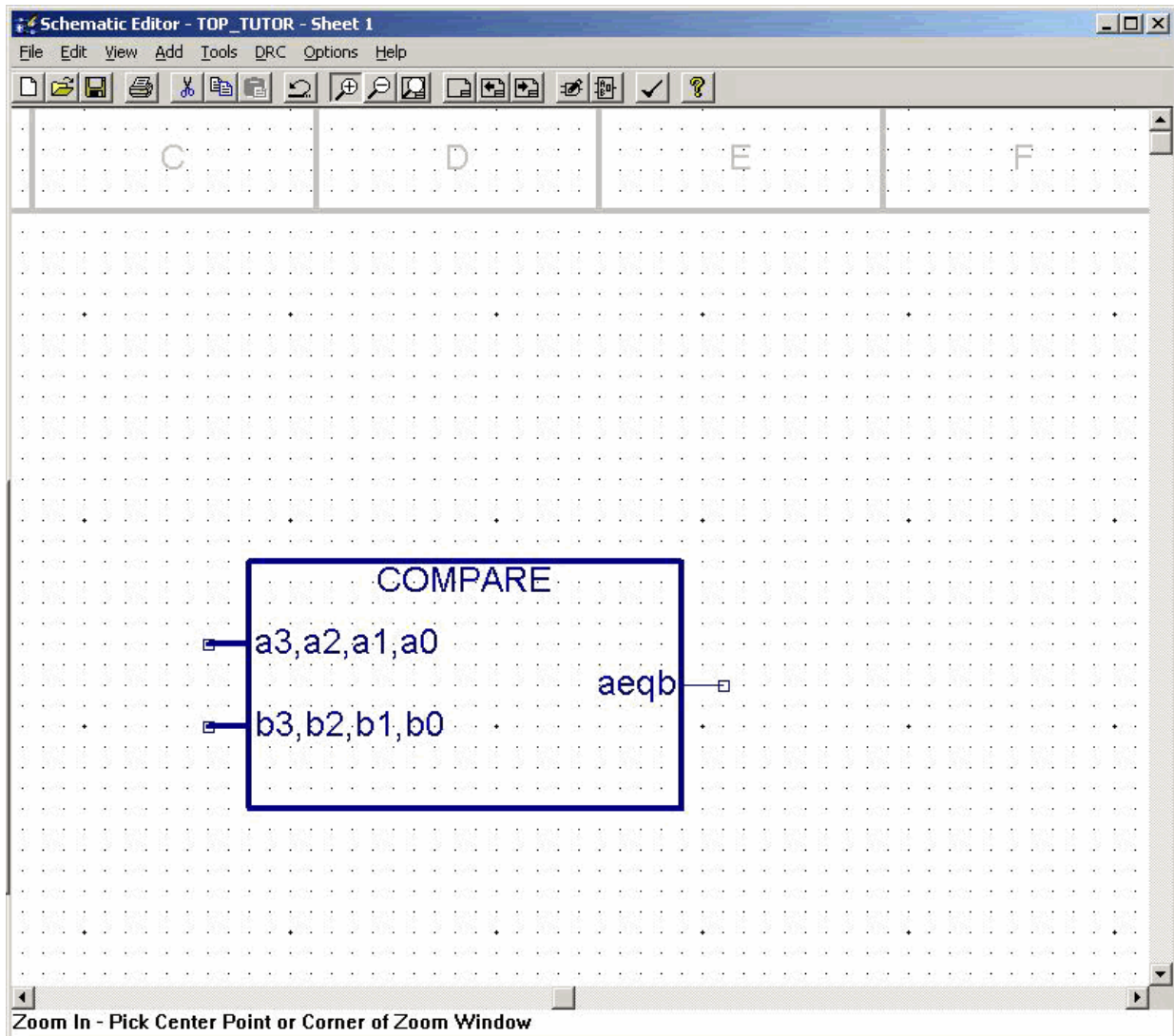
1. In the Symbol Libraries dialog box, under Library, select **generic\regs.lib**.
2. Under Symbol, select **G_DC**.

- Place the symbol so that the **G_DC** symbol's output is aligned with the **b** input pin on the COMPARE block symbol. (Ignore the title block in the schematic.)



- Choose **View > Zoom In**. The cursor changes to a large Z.

- Click a portion of the schematic to enlarge it to a readable size. Or drag an area to zoom in to a view similar to the following one.



- Right-click to cancel the zoom command.

Task 8: Place Symbols from the IOPAD Generic Symbol Library

In this step, you will continue adding symbols to the schematic. However, you will use the IOPADS generic symbol library.

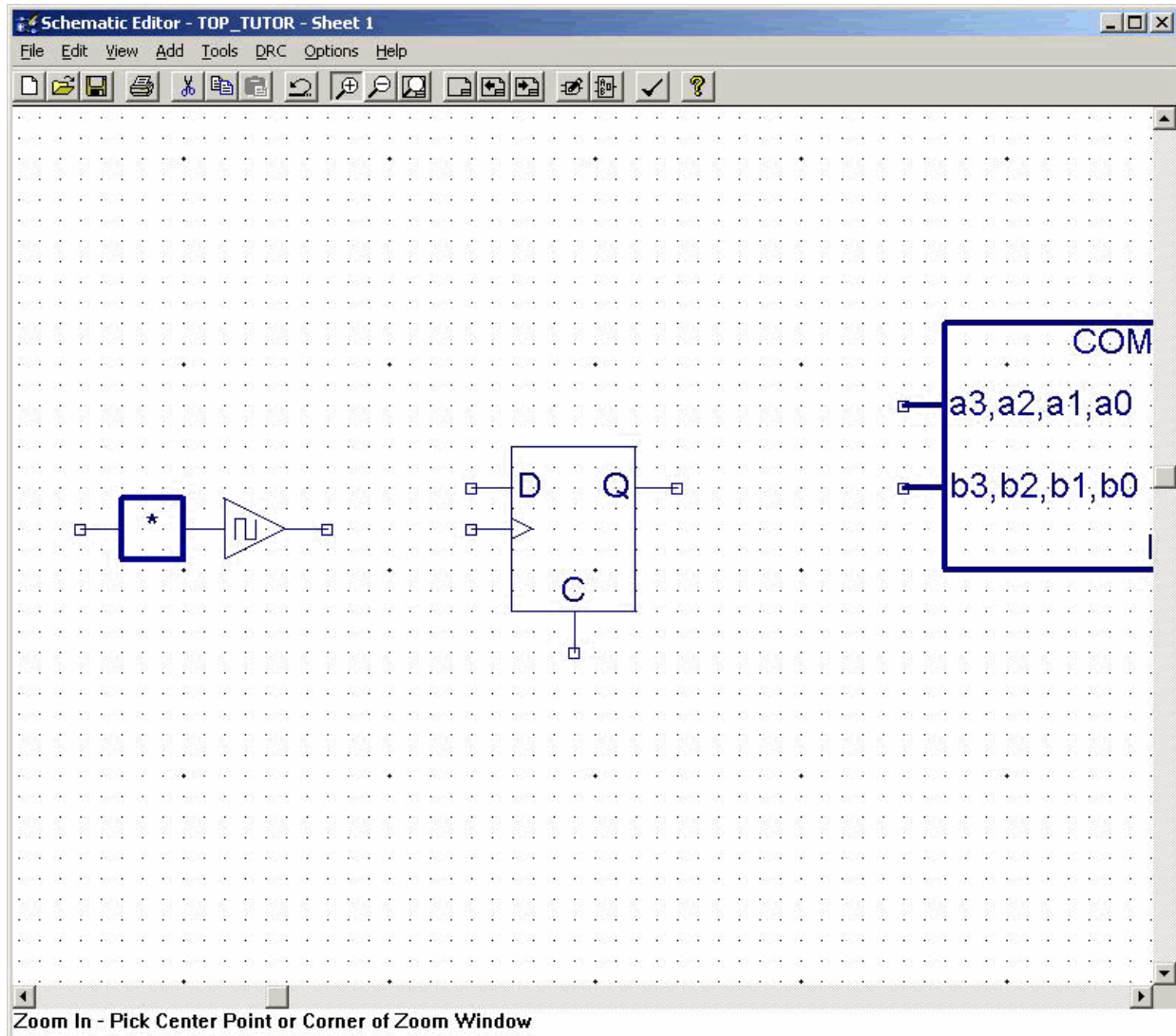
Note

You can turn off the grid display by choosing **Options > Preferences**.

To add a clock buffer symbol:

- In the Symbol Libraries dialog box, under Library, select **genericiopads.lib**.

2. Under Symbol, select **G_CLKBUF**.
3. Place the symbol so that the output is aligned with the clk input pin on the G_DC symbol.



Note

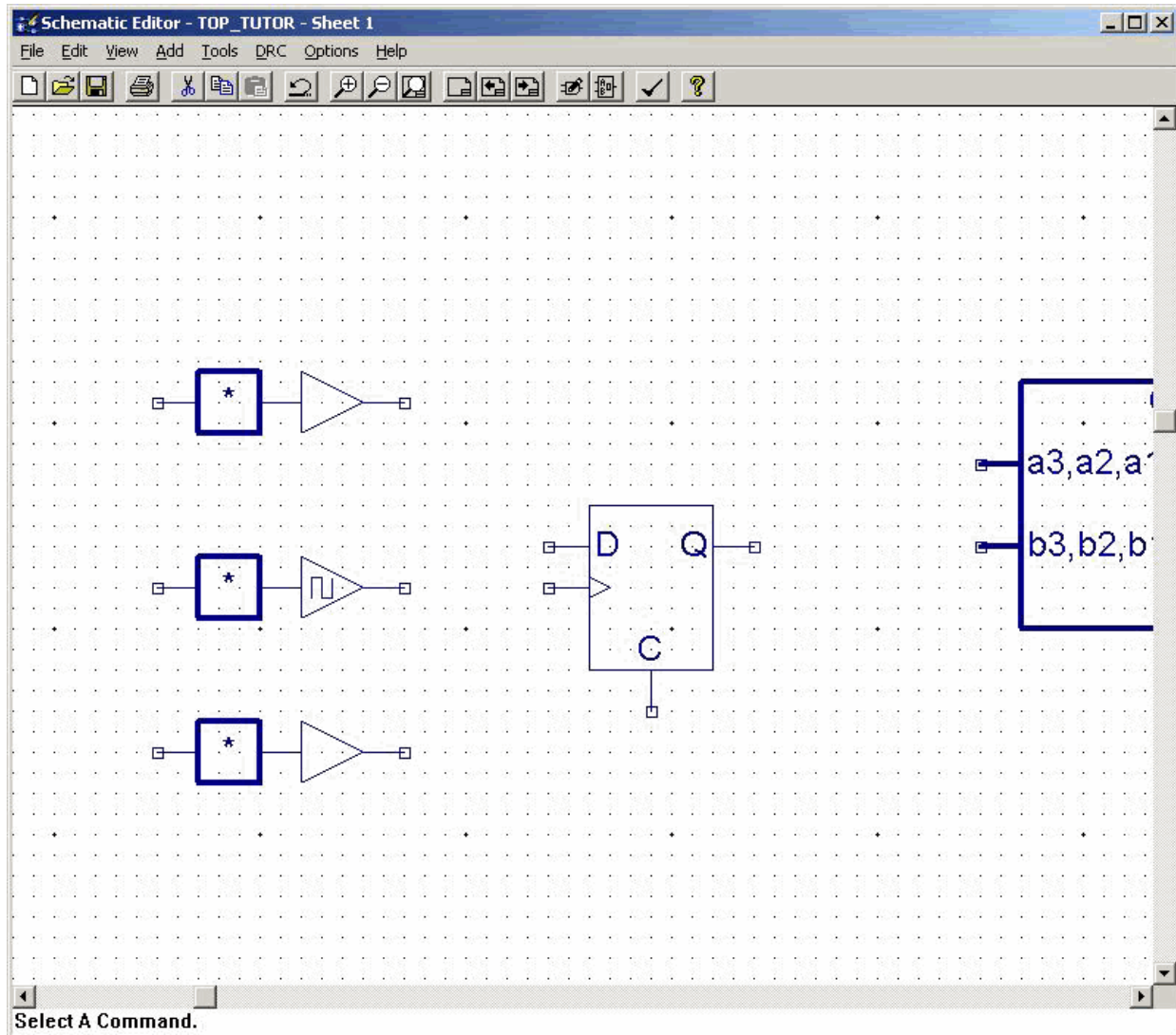
The spacing of symbols in the schematic is very important. If symbols are placed too close to each other, you will not be able to add wires, bus taps, and so forth between the symbols. In the following step, make sure the horizontal distance between the G_CLKBUF symbol and G_DC symbol is equal to at least the width of the G_CLKBUF symbol (see the above picture).

If you make a mistake placing a symbol, you can choose Edit > Cut or Move, and then click the symbol that is to receive the action.

Now you will add two input buffer symbols.

4. Under Symbol, select **G_INPUT**.

- Click once to place a symbol instance. (Try to align symbols vertically, as shown in the picture.)



- Click once again to place another symbol instance.

Task 9: Add Wires and Buses

The next step is adding wires and buses to interconnect the symbols. You draw wires and buses the same way; the Schematic Editor knows if a wire is a bus or a single net by the wire's name. In this task, you will draw all of the wires. Later, you will add names to the wires.

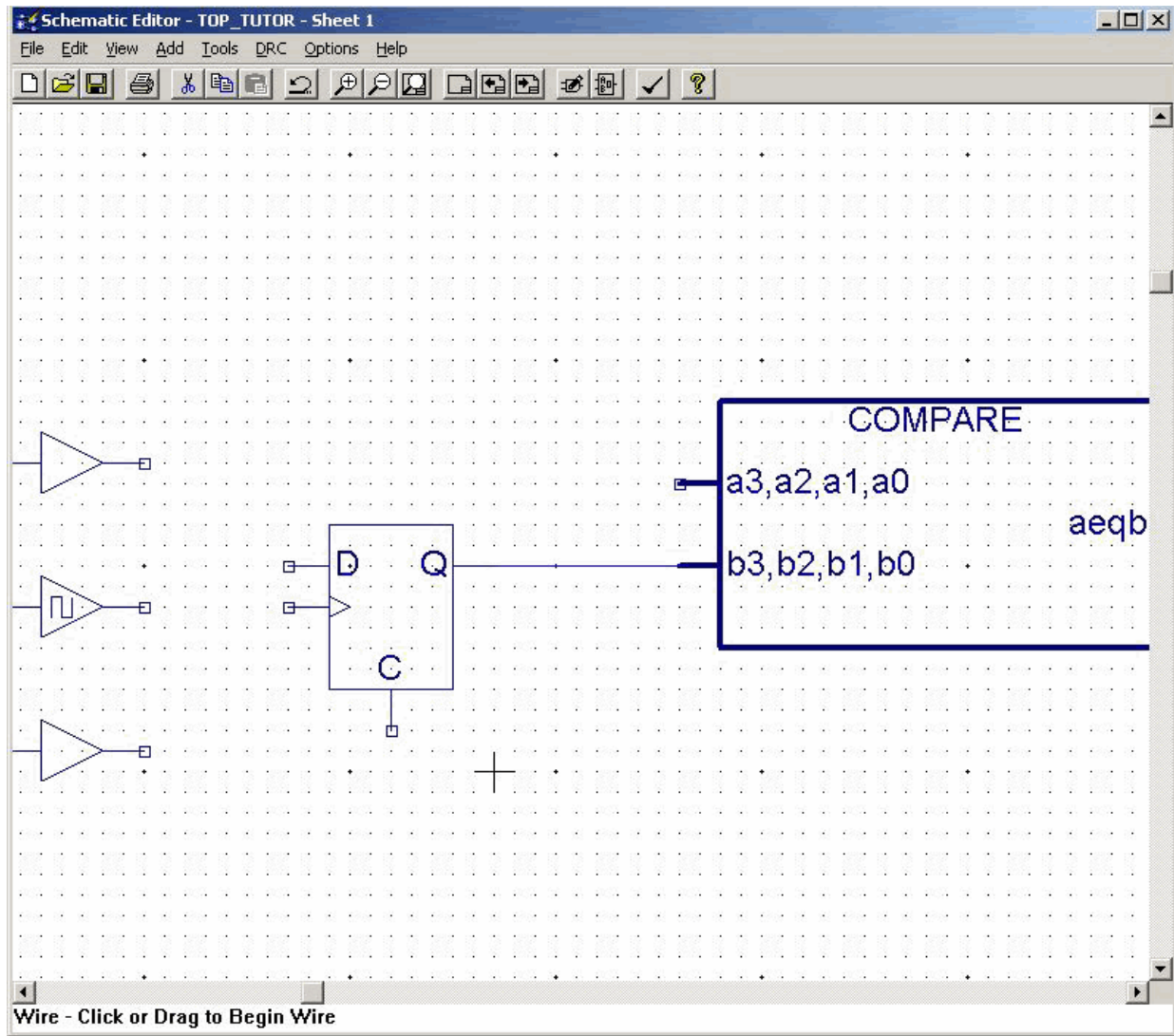
Note

The Add Symbol command from the previous task is canceled when you select the command for this new task.

To connect the flip-flop output to the COMPARE symbol:

- Choose **Add > Wire**.

2. Click the **Q** output of the flip-flop.
3. Drag the wire across to **COMPARE** and click the **b** pin to end the wire segment.



Task 10: Edit the Schematic

It is easy to make changes to the schematic when you need to erase or reposition any of the wires or symbols.

To erase a symbol or wire:

1. In the Schematic Editor, choose **Edit > Delete**.
2. Click one of the input buffer symbols to erase it. That item disappears.

3. Now choose **Edit > Undo**. The buffer symbol reappears.

Note

If you erase or reposition the wrong item, you can always undo your action by choosing **Edit > Undo**.

To reposition a symbol:

1. Choose **Edit > Move**.
2. Click the **COMPARE** block. The symbol is attached to the cursor.
3. Move the cursor toward the bottom of the schematic and then click to place the block.
4. Choose **Edit > Undo** to return the block to its previous position.

To reposition several items at once:

1. Choose **Edit > Drag**.
2. Drag a square around both the **COMPARE** block and the flip-flop. Both items are attached to the cursor.
3. Move the cursor to the bottom portion of the schematic and click to place the items.
4. Choose **Edit > Undo** to return the items to their previous positions.

Note

Almost all commands remain in effect until you select a different command. For example, if the Add Wire command is active, you can continue to draw wires until you select a different command.

Task 11: Add Wires to Connect the Symbols

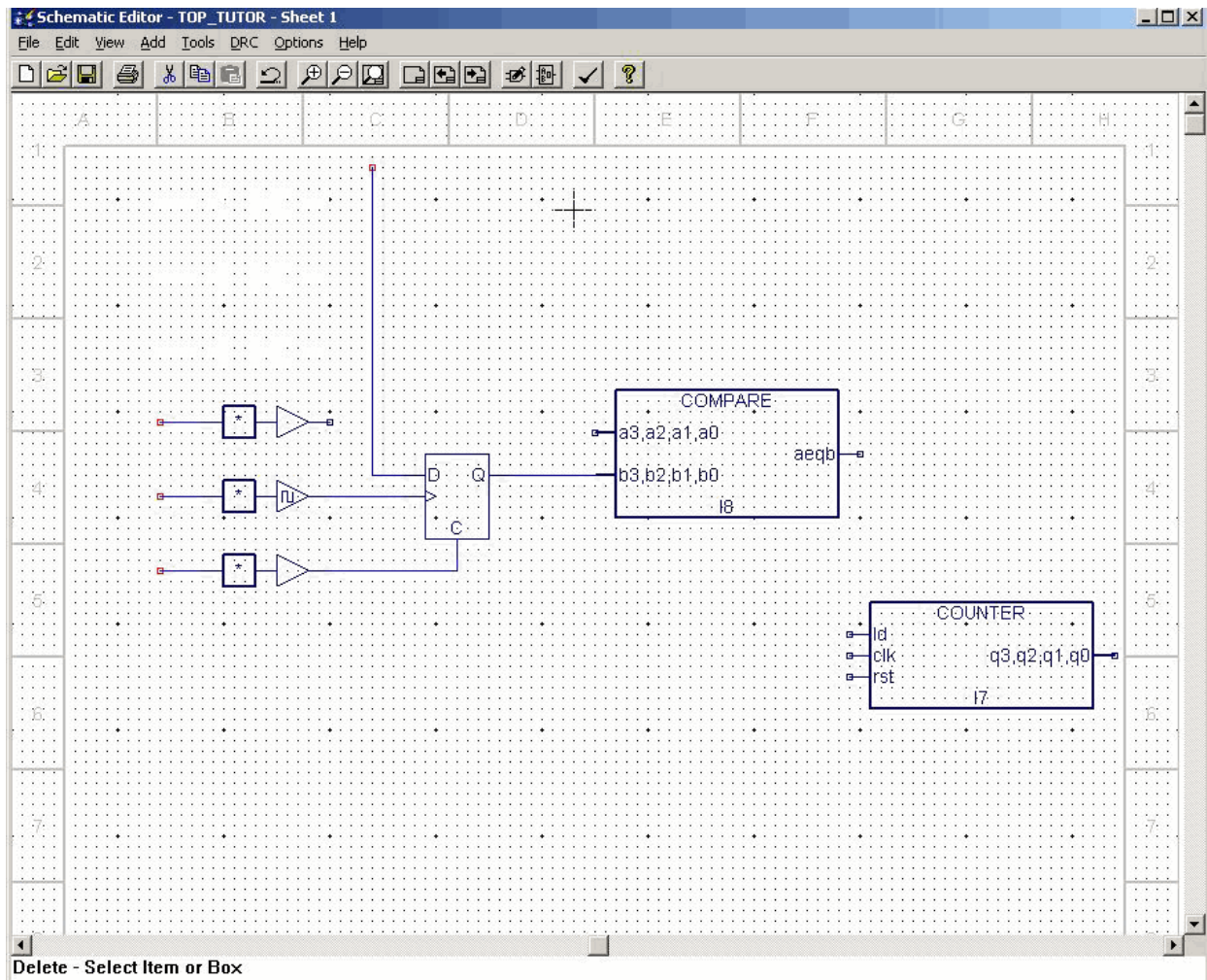
In this step, you will add three wires to the flip-flop. Also, you will add wire stubs to the buffers.

The length of the horizontal and vertical segments of the D input wire (step 7 below) is important. The wire segments must be long enough so that you have room to add a bus name to the horizontal wire segment and add bus taps between the vertical wire segment and input buffers.

To add a wire from D input:

1. Choose **Add > Wire**.
2. Click the **D input**. Move the cursor to the left to create a horizontal segment with a length equal to about half the horizontal distance between the input buffer and flip-flop. Click once to end the segment.
3. Move the cursor up until the vertical height of the wire is about equal to three times the height of the flip-flop.
4. Right-click to end the wire.
5. Add a wire from the **C input** to the **input buffer**.

6. Add a wire from the **clock input** to the **clock buffer**.
7. Add wire stubs (short wires) to the **input buffers** and the **clock buffer**. (Double-click to end a wire in space, or right-click to cancel the command.)



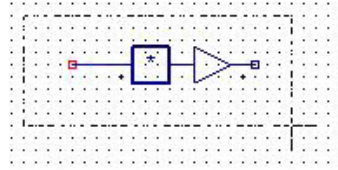
Task 12: Duplicate the Input Pad and Wire Stub

This step shows you how to use the Duplicate command to quickly add the rest of the input pads.

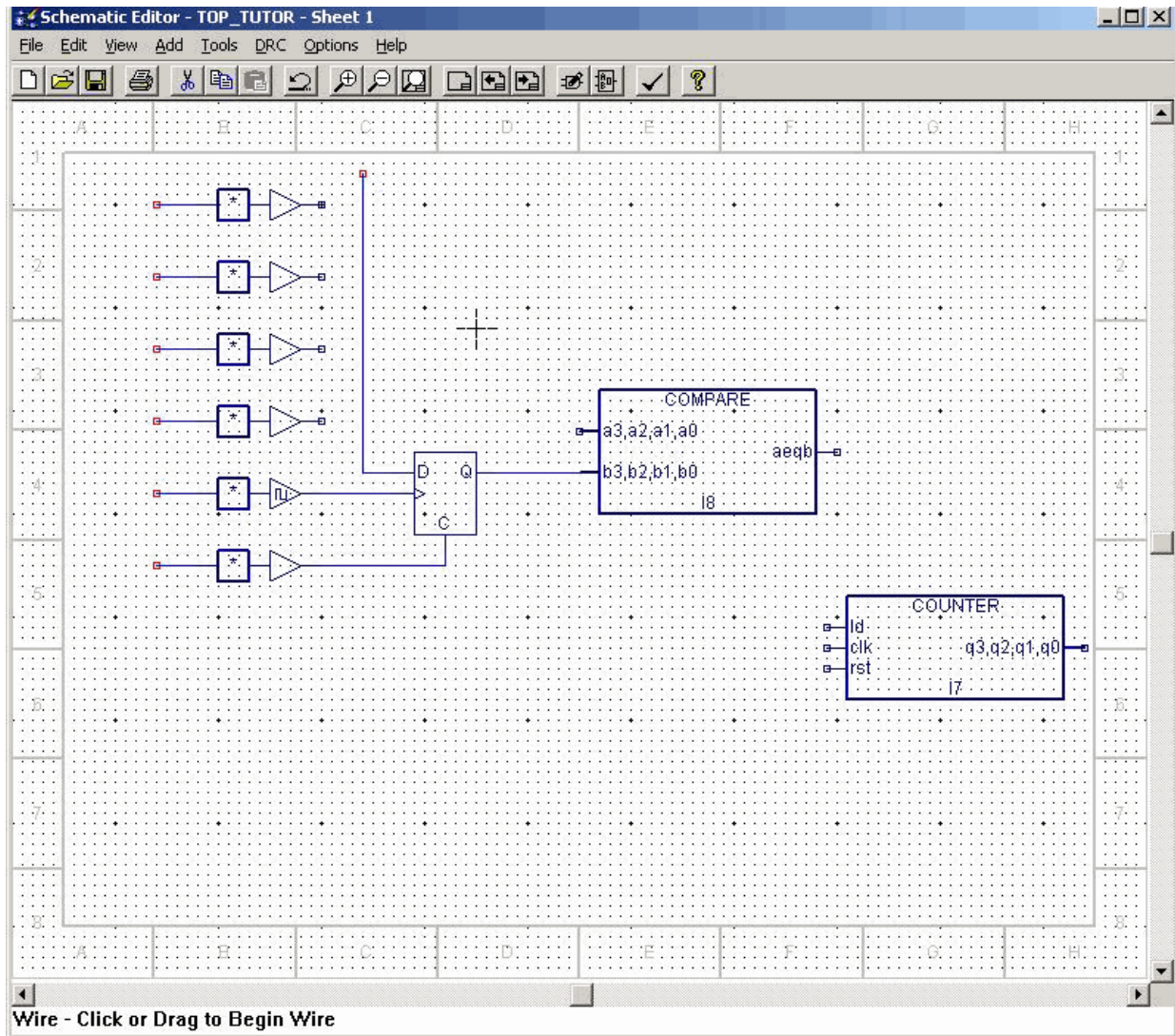
The Duplicate command lets you copy one or more elements, and then place them at different locations within the same symbol or schematic. You can place the duplicated item as many times as you want until you select another command. Duplicate differs from the Copy/Paste command sequence only in that it does not change the contents of the clipboard.

To duplicate the input buffer:

1. Choose **Edit > Duplicate**.
2. Hold down the mouse button and drag a region around the input buffer and wire stub.



- Click to place three of the duplicated buffer and wires as shown.



- Right-click to cancel the command.

Task 13: Name the Buses

Any single- or multi-wire connection between pins is called a network, or net. A bus is a combination of two or more signals in a single wire. Buses are a convenient way to group related signals. This grouping can produce a less cluttered, functionally clearer drawing and clarify the connection between the main circuit and a block symbol.

There are two types of buses: ordered and unordered. An ordered bus has a compound name consisting of the names of the signals that comprise the bus. Any signals can be combined into an ordered bus, whether they are related or not.

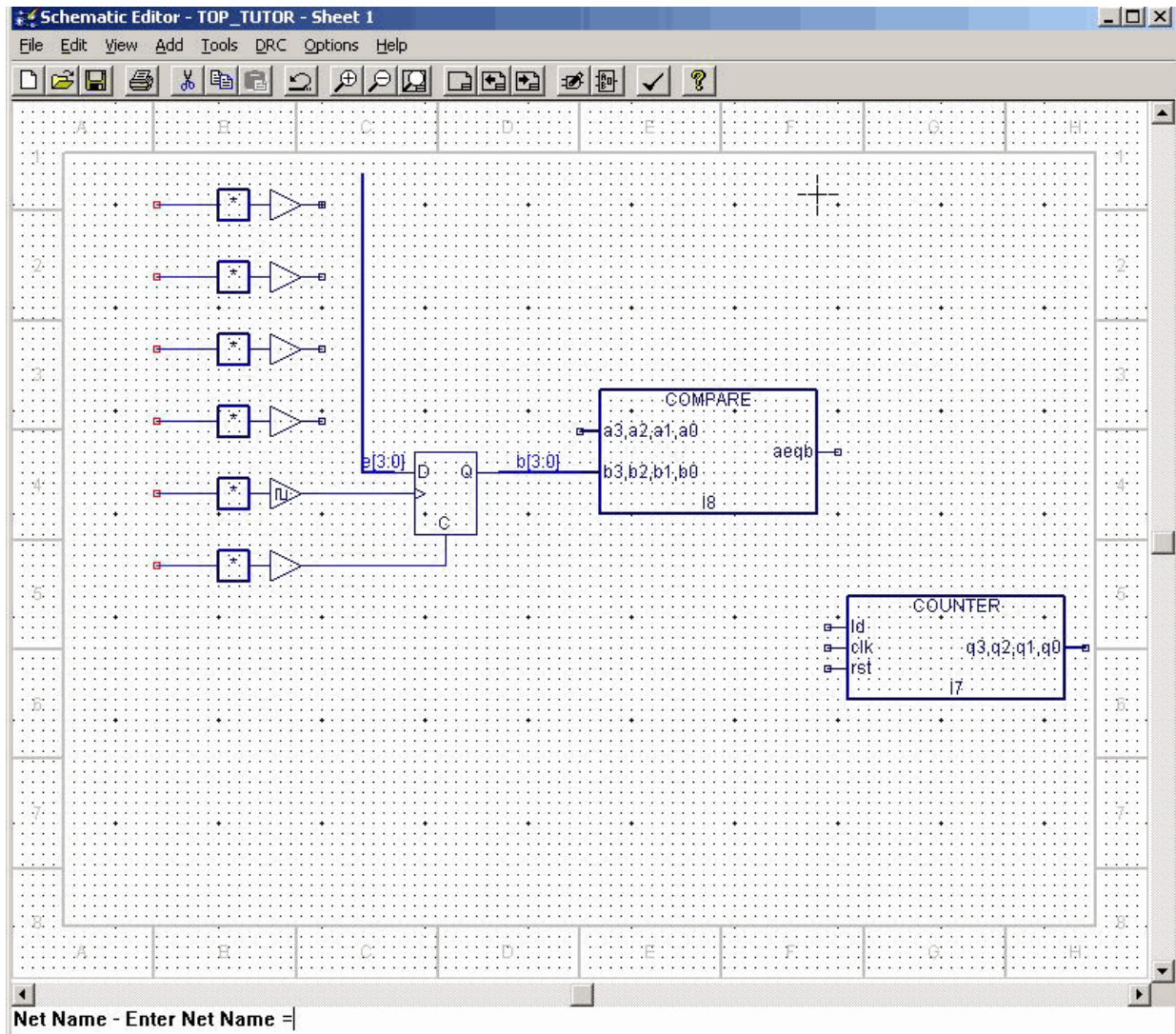
A net becomes an ordered bus when it is given a compound name. You form a compound name by adding a sequence of numbers to the name. The sequence is specified as a starting number, an ending number, and an optional increment (default = 1). The numbers are positive integers and are delimited by commas (,), dashes (-), or colons (:). The sequence is enclosed in brackets [], parentheses (), or curly braces { }.

In this step, you will name two buses.

To name the buses:

1. Choose **Add > Net Name**.
2. On the Prompt Line at the bottom of the Schematic Editor, type the name of the bus, **b[3:0]**, and then press **Enter**. The name is attached to the cursor.

- Click the wire between the flip-flop and the COMPARE symbol to place the name on the wire. Notice that the wire thickens once you have named it a bus.



- Now name another bus by typing on the Prompt Line **e[3:0]**, and then press **Enter**.
- Click the horizontal segment on the left side of the flip-flop. Again, the wire thickens and now carries the name of the bus.

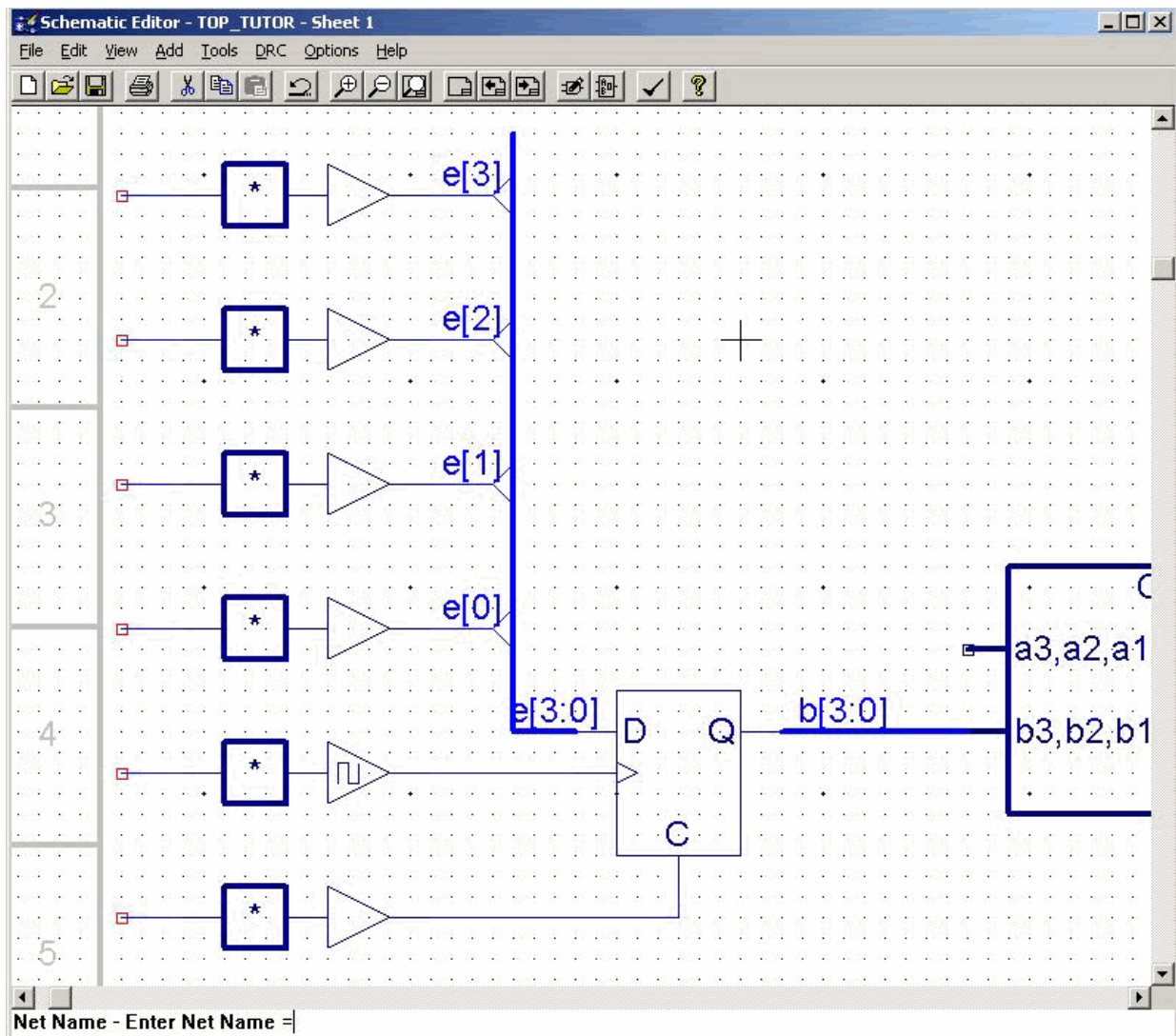
Task 14: Add Bus Taps with Signal Names

Signals enter and exit a bus at points called bus taps. A bus tap can be added to any existing bus, net, or wire. If a net or wire is not already a bus, adding the tap automatically promotes it to a bus. You can add bus taps only on vertical or horizontal sections of a bus. Tap connections are shown with two diagonal lines, rather than a solder dot.

There are several ways to add bus taps. The following procedure shows you how to create a tap, the connecting wire, and the net name in one simple step using the Net Name command.

To add bus taps with signal names:

1. Choose **Add > Net Name**.
2. Click the bus labeled **e[3:0]**. The cursor picks up the name of the bus.
3. Right-click once to split the bus name into its individual signal names. The signal name **e[3]** is attached to the cursor.
4. In one action, click the pin of the top input pad, drag a wire to the bus, and then release the mouse button. The software adds a bus tap, wire, and signal label. Notice that the signal name decrements.



5. Add the remaining taps in descending order just by clicking the pins of the buffers.

Task 15: Add Input Net Names

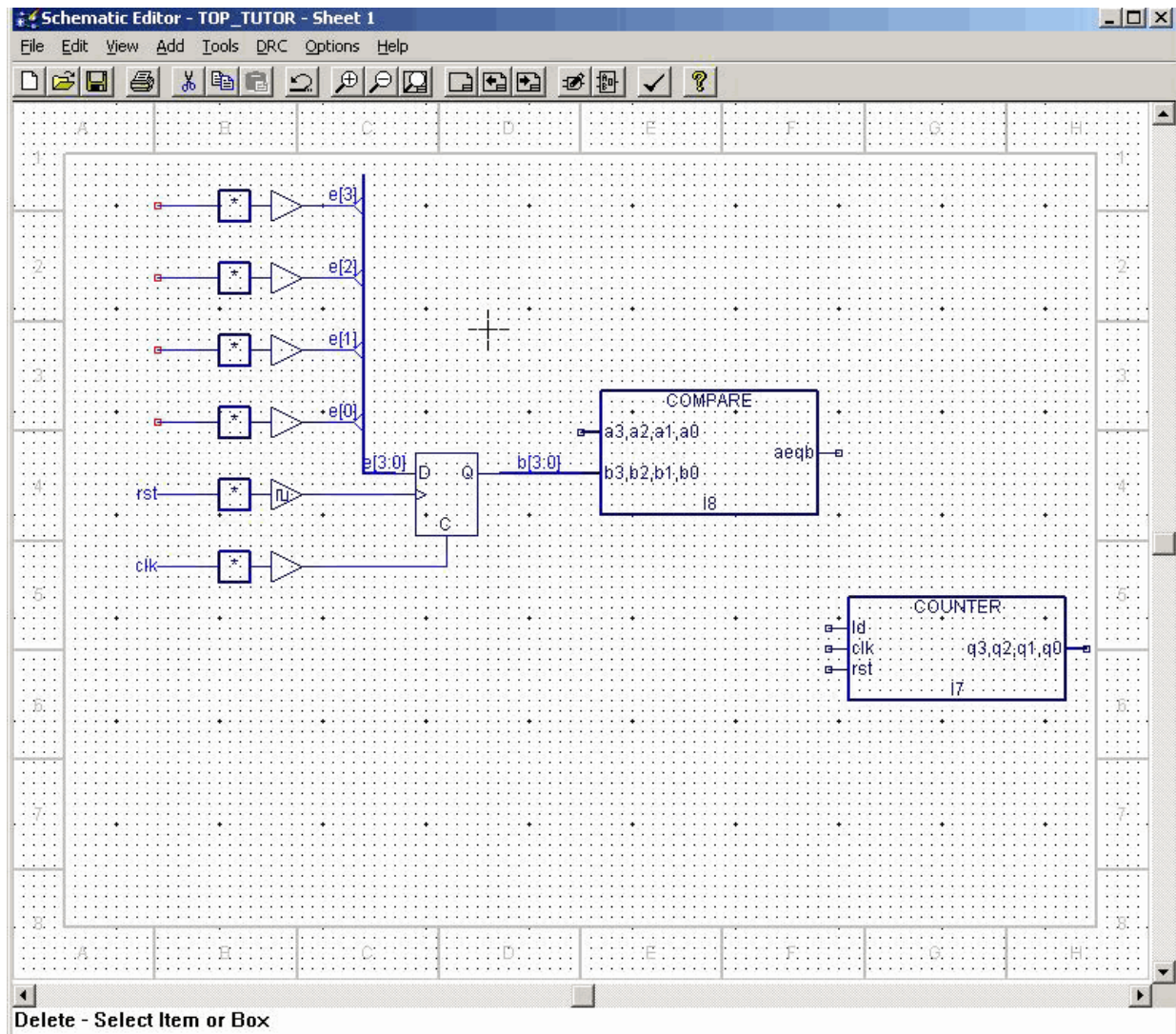
Every net has a name, either assigned by you or by the Schematic Editor. You can override any name assigned by the Schematic Editor by assigning one of your own using the Net Name command.

You can name nets one at a time. A faster way is to create a compound name (in this example, a group of unique names), and then sequentially attach individual names of a compound name to different nets.

To add a compound net name:

1. Choose **Add > Net Name**.
2. On the Prompt Line, type: **clk,rst**, and then press **Enter**. (Do not forget the comma separating the names.)
3. Notice that both names are attached to the cursor. Right-click once to separate the names. Now only clk is showing.

- Click the end of the input wire for the clock buffer. The software adds the first name (clk) to the wire. Also notice that the next name in the list (rst) appears on the cursor.



- Click the end of the rst input buffer to add the final name (rst) to the wire.

Task 16: Add Data Input Net Names

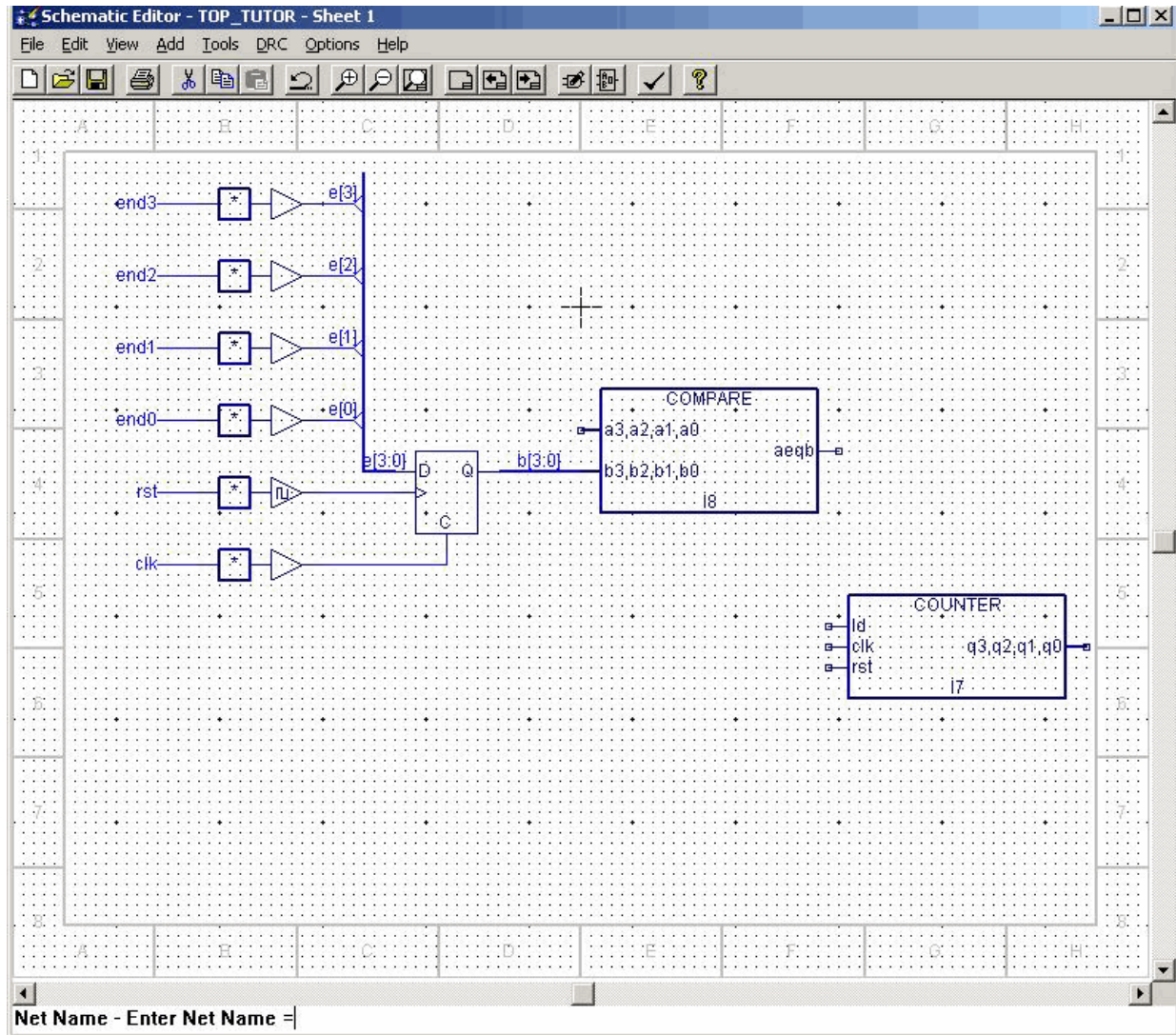
Another net naming feature makes it easy to deal with buses by automatically incrementing the net name as you place it. In this step, you will add net names to the end segments, and the Schematic Editor will automatically increment the name.

To add sequential net names:

- Choose **Add > Net Name**.
- On the Prompt Line, type: **end0+**, and then press **Enter**. The name end0 is attached to the cursor.

This tells the Schematic Editor to name the signal end, to start numbering at 0, and to increment (+) the numbers as the names are placed.

3. Click the bottom data input wire stub. The Schematic Editor places the name end0 and automatically increments the net name.



4. Click the next wire stub. Repeat this step until all data input wire stubs have net names.
5. Right-click when end4 appears on the cursor to cancel the command.

Task 17: Create Iterated Instances of the Flip-Flop

A powerful feature in the schematic is its capability of using iterated instances, which allow a single symbol to represent multiple instances connected in parallel.

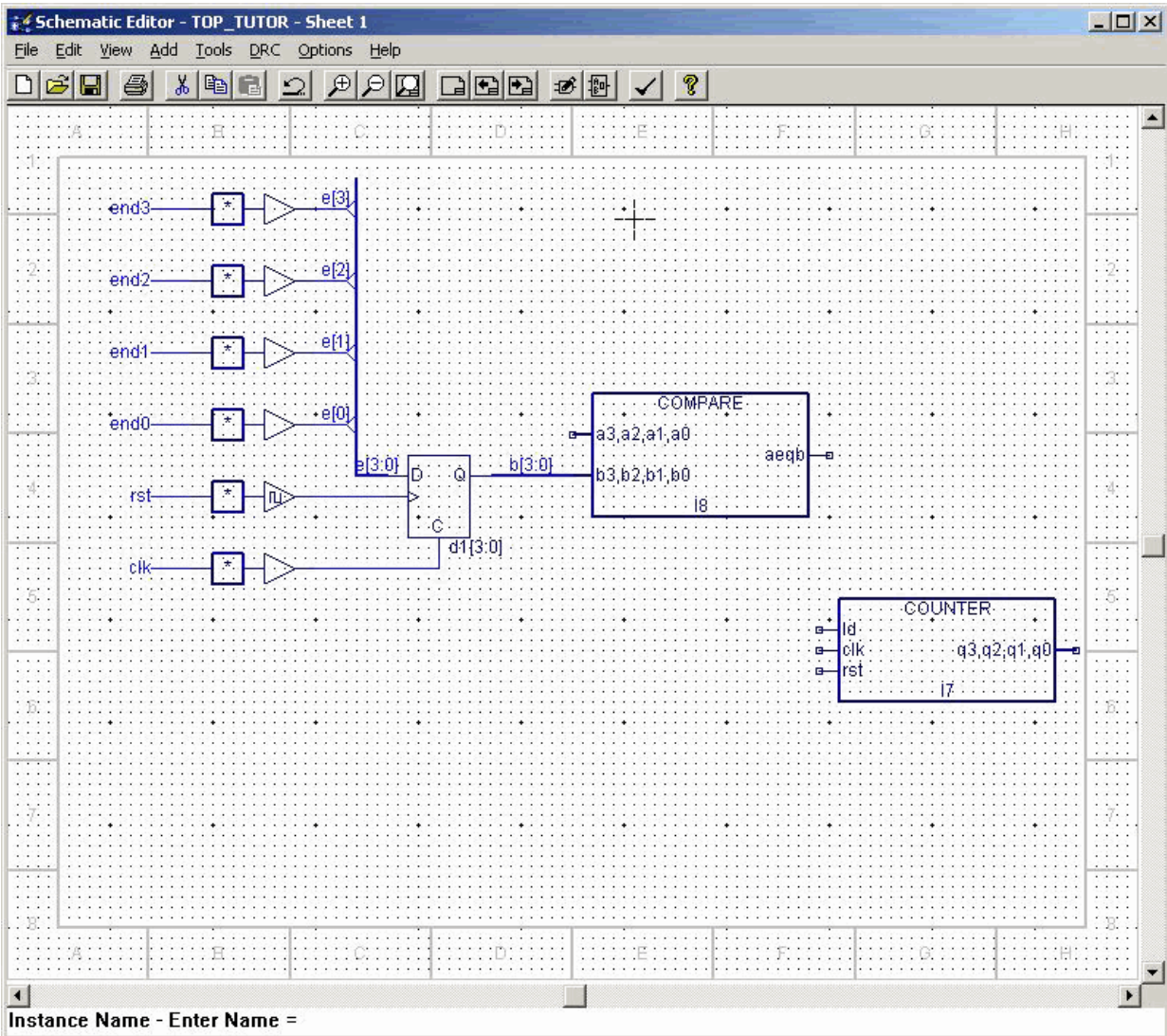
You can convert a single instance into an iterated instance by giving it a compound instance name of the following form:

INV[3-10]

In this case, eight instances of the symbol you have named INV are created, but the symbol appears only once in the schematic.

To create iterated instances of the flip-flop:

1. Choose **Add > Instance Name**.
2. On the Prompt Line, type: **d1[3:0]**, and then press **Enter**. The name is attached to the cursor.
3. Click the flip-flop once. The Schematic Editor places the label on the symbol.



The flip-flop is now really four flip-flops, with signal e[3] feeding d[3], e[2] feeding d[2], and so forth. The common signals are connected in parallel to the common pins, such as clk and rst.

Task 18: Add Input Markers

An I/O marker is a special indicator that identifies a net name as a device input, output, or bidirectional signal. It establishes net polarity (direction of signal flow) and indicates that the net is externally accessible.

The Schematic Editor Consistency Check command uses I/O markers to flag any discrepancies in the polarity of marked signals and the symbol pins. Discrepancies in polarity are also flagged each time that you run the Hierarchy Navigator.

To add input markers:

1. Choose **Add > I/O Marker** to open the dialog box.
2. Select **Input**.



3. You can add a marker by clicking at the point where the I/O marker touches the end of a horizontal or vertical wire segment or bus. However, if you have a group of nets to which you want to add markers, there is a faster way. Select all of the input wires at once by dragging a region around them. Add the net name to the port, then choose **Add > I/O Marker**. Drag a square to cover that port. The Schematic Editor adds markers to all the nets at once.

The following tasks are covered in this lesson:

- ◆ Task 19: Import the Completed Schematic
- ◆ Task 20: Create a Matching Symbol
- ◆ Task 21: Check the Schematic for Consistency Errors

Task 19: Import the Completed Schematic

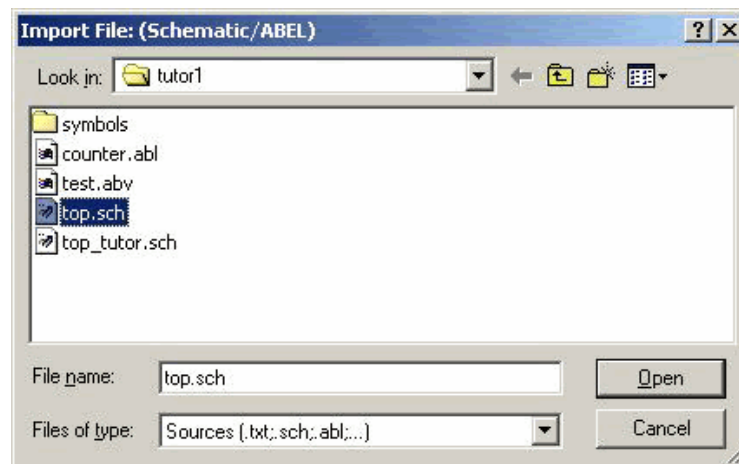
In this step, you will remove the schematic source you have been working on and import the completed schematic of the same name.

Note

If you skipped Lesson 2, you do not have to complete steps 1 and 2.

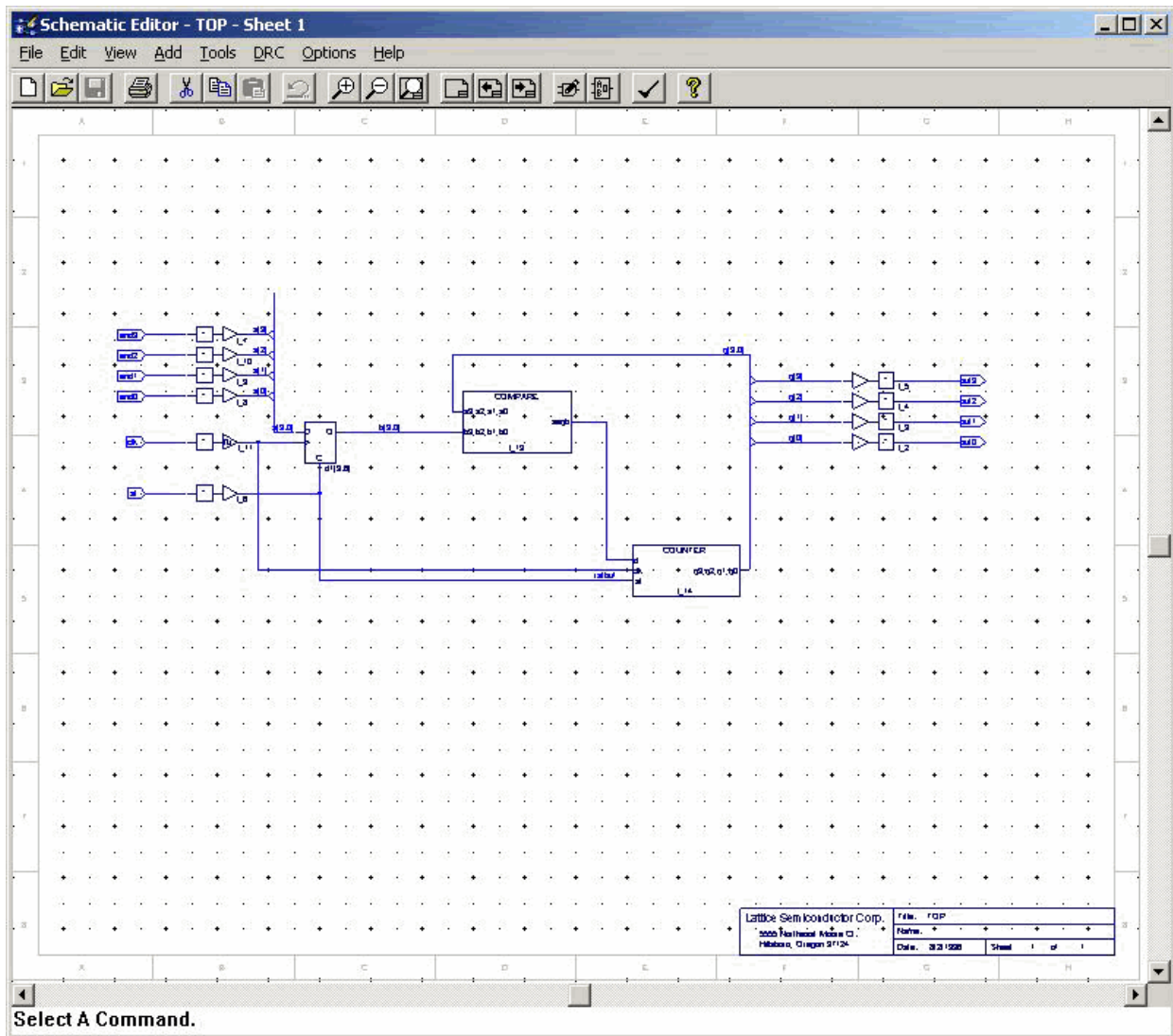
To import the completed schematic:

1. In the Sources in Project window of the Project Navigator, select the **top_tutor** schematic source.
2. Choose **Source > Remove** to delete the source from the project.
3. Choose **Source > Import** to open the Import File dialog box.
4. Select **top.sch** and click **Open**.



5. Double-click the top source to open the schematic in the Schematic Editor.

The Schematic Editor opens, showing the completed schematic.



Task 20: Create a Matching Symbol

You can use the Matching Symbol command to create a symbol file (*.sym) for the schematic currently loaded, with the same base name. The input and output pins on the symbol have the same signal names and polarities as the I/O markers in the schematic.

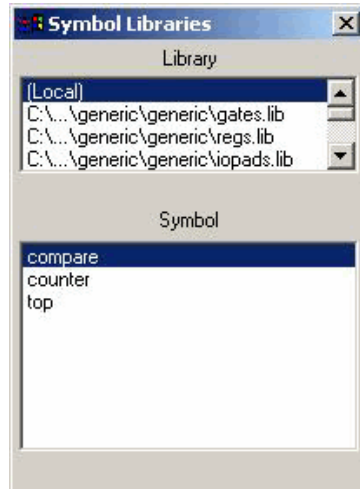
The Schematic Editor creates the symbol in the same directory as the schematic. You can use the Add Symbol command to insert the symbol into any other schematic.

In this step, you will create a symbol for the top schematic. The symbol will be saved in the Local symbol library in the project directory.

To create a matching symbol for the Top schematic:

1. Choose **File > Matching Symbol**. The Schematic Editor automatically creates a symbol.

2. Choose **Add > Symbol** to open the Symbol Libraries dialog box.
3. Under Library, scroll to the top and select **(Local)**. Notice, without selecting it, the symbol named top.



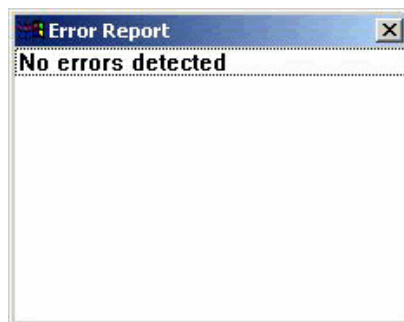
4. Close the Symbol Libraries dialog box.

Task 21: Check the Schematic for Consistency Errors

The Schematic Editor continually checks for errors, such as closed loops and shorted nets, while you are drawing your schematic. You can also check your schematic for other errors such as unconnected wires or pins or an unnamed signal tapped from a bus. Errors found are shown in a “hot” list box. Clicking an error causes the cursor to jump to its location.

To check for consistency errors:

1. Choose **DRC > Consistency Check** to open the Error Report. There should be no errors in the report.



2. Close the Error Report.
3. Choose **Edit > Delete**.
4. Drag the area around the circuitry on the four gates on the far right side of the schematic, and then release the mouse button.
5. Choose **DRC > Consistency Check** again.



6. In the Error Report, select an error. Notice that the schematic view shifts to the location of the selected error.
7. Close the Error Report.
8. Choose **File > Exit** to exit the schematic. When asked to save your changes, click **NO**.

You have just created and checked a top-level schematic. Now you will create an ABEL-HDL source and add it to the project.

Lesson 4: Adding ABEL-HDL Sources to the Project

The Project Navigator now lists the schematic (top) and the two block symbols, counter and compare, referenced in the schematic. These sources have the undefined icon next to them because they do not exist yet. Remember that this is a top-down design example.

This lesson leads you through the steps necessary to add two lower-level ABEL-HDL modules to the project. One you will create from scratch. The other you will import.

The following tasks are covered in this lesson:

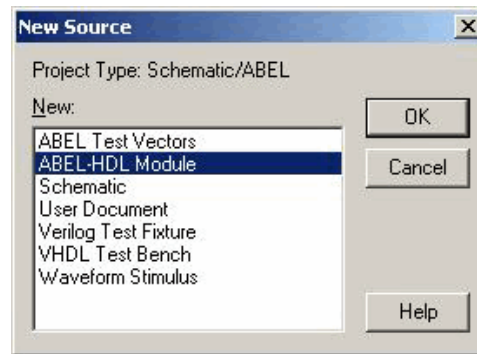
- ◆ Task 22: Create a New ABEL-HDL Source File Template
- ◆ Task 23: Enter the ABEL-HDL Source Description
- ◆ Task 24: Import an Existing ABEL-HDL Source File

Task 22: Create a New ABEL-HDL Source File Template

Creating a new ABEL-HDL source file involves two steps. First, you create the template. Then you enter the source description. In this task, you will create a template for the Compare ABEL-HDL module.

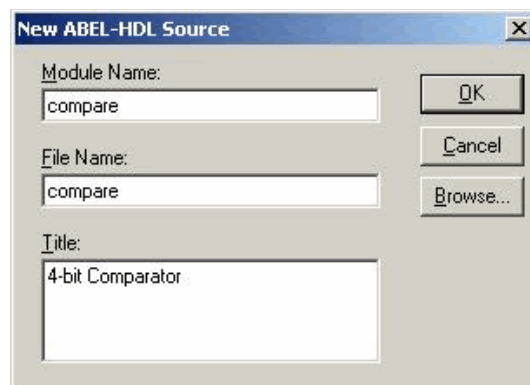
To create an ABEL-HDL template:

1. In the Project Navigator Sources in Project window, double-click **COMPARE** to open the New Source dialog box.
2. Select **ABEL-HDL Module** and click **OK**.



The Text Editor opens, along with the New ABEL-HDL Source dialog box that prompts you for module information.

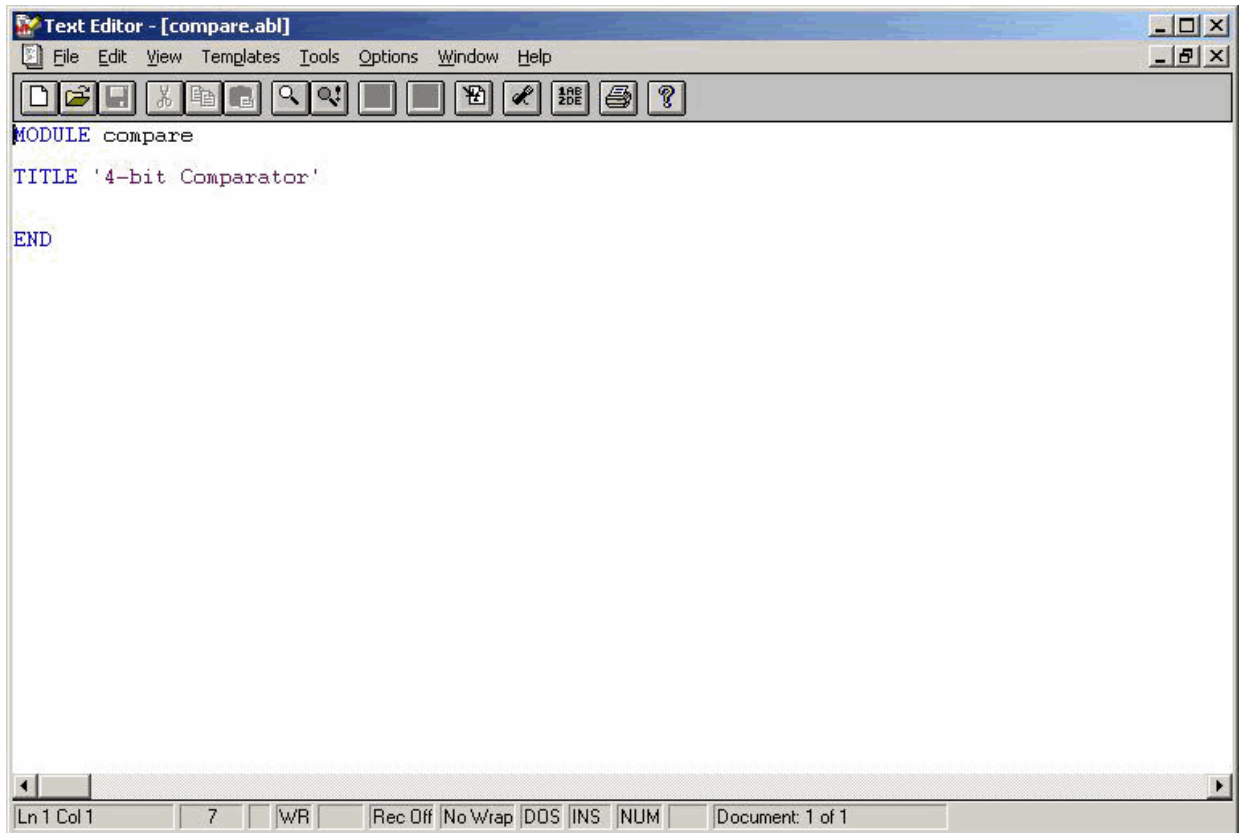
3. In the dialog box, type the information shown in the following illustration.



The text is case-sensitive. Make sure that you enter the text as shown. The Module Name matches the corresponding block symbol name. The module and file names do not have to be the same, but it makes things simpler.

4. Click **OK** to close the dialog box.

The Text Editor appears with an ABEL-HDL template. The template shows the module and file names that you entered.



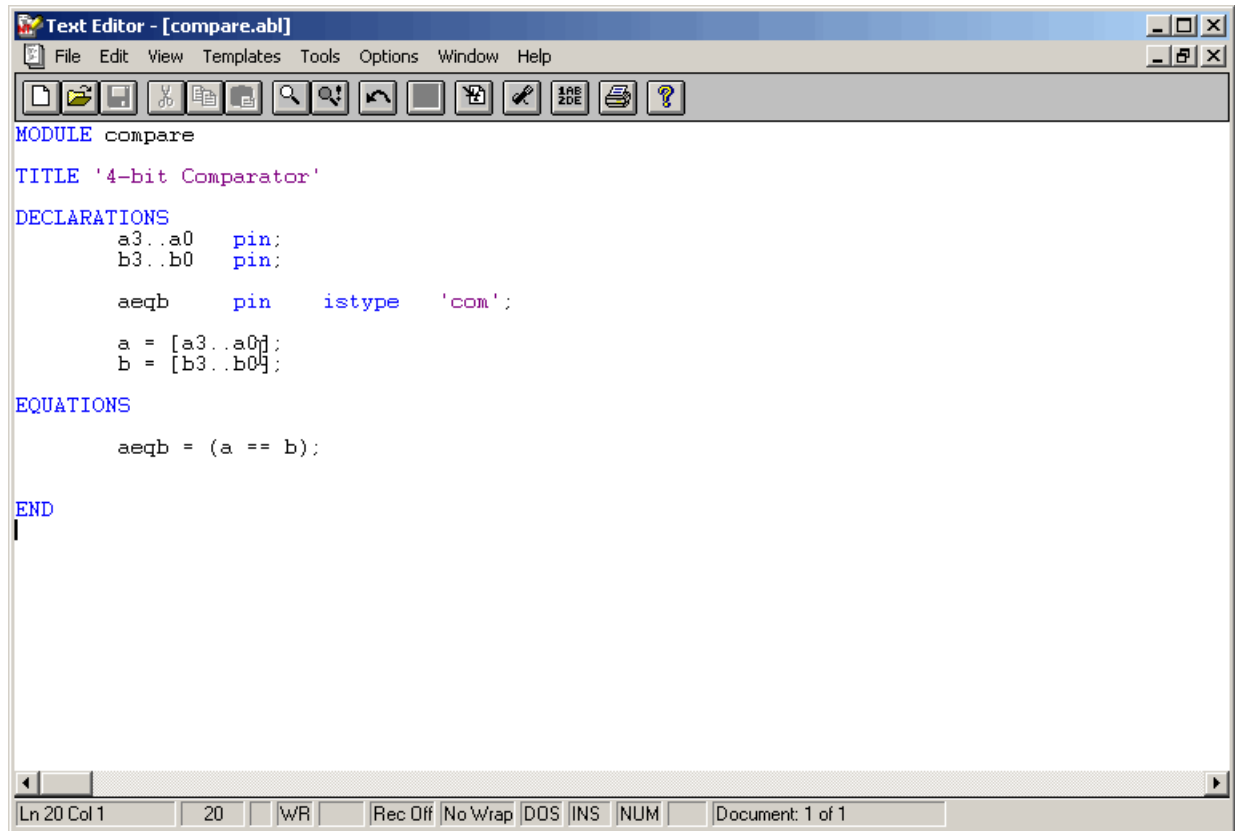
Task 23: Enter the ABEL-HDL Source Description

In this step, you will complete the ABEL-HDL module by entering its description.

To describe an ABEL-HDL module:

1. In the Text Editor, type the text description as shown in the following picture.

Notice that the ABEL-HDL pin declarations must match the name and the case of the I/O pins on the corresponding block symbol.



```
Text Editor - [compare.abl]
File Edit View Templates Tools Options Window Help
MODULE compare
TITLE '4-bit Comparator'
DECLARATIONS
    a3..a0    pin;
    b3..b0    pin;

    aeqb     pin    istype 'com';

    a = [a3..a0];
    b = [b3..b0];
EQUATIONS
    aeqb = (a == b);
END
```

Ln 20 Col 1 20 |WR |Rec Off |No Wrap |DOS |INS |NUM | Document: 1 of 1

2. When you are through, choose **File > Save** to save the file.
3. Close the Text Editor.

In the Project Navigator Sources window, notice that the icon next to the compare file has changed, and that the file name (compare.abl) appears next to the name. This indicates that the file is now a defined module.

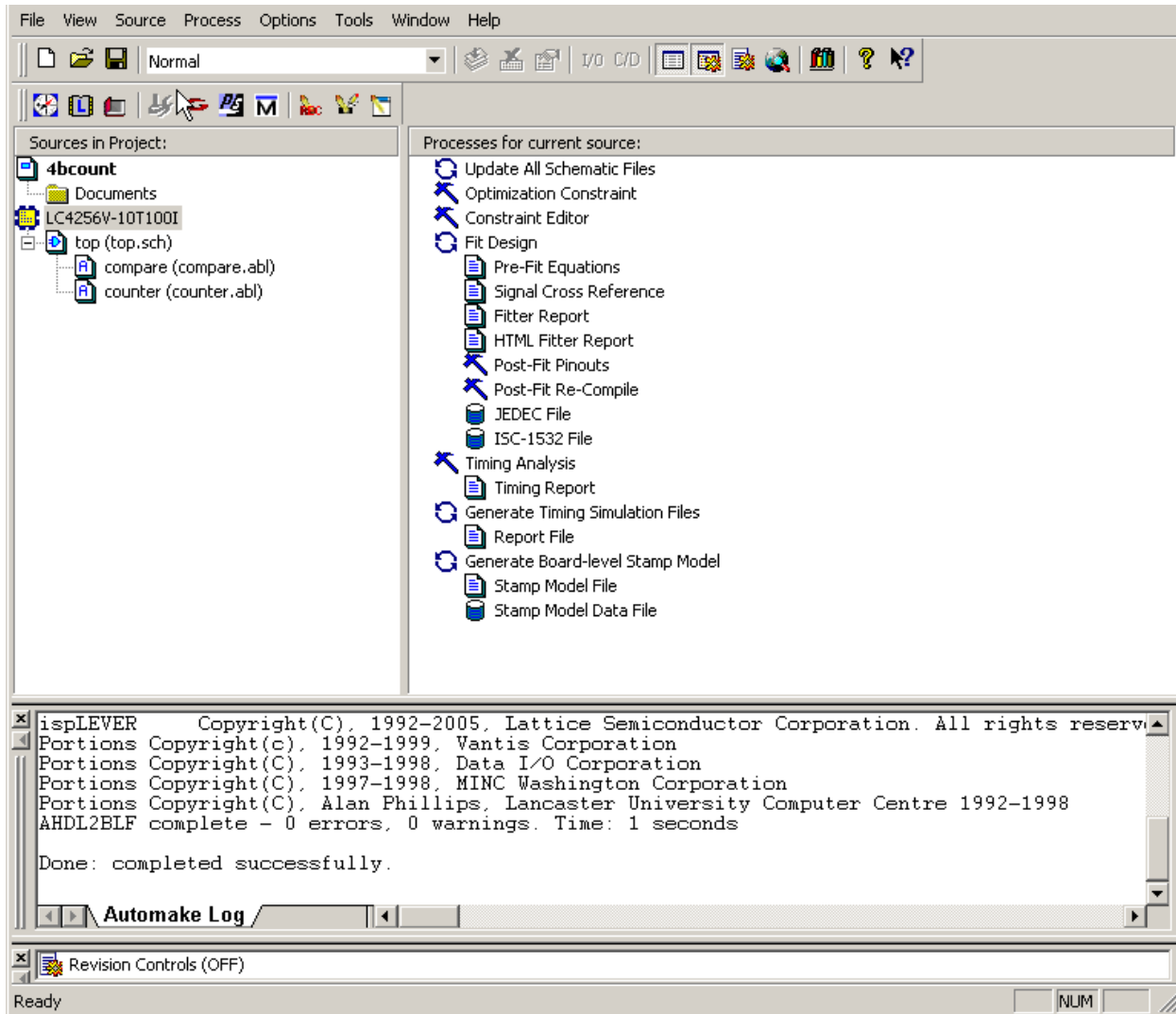
Task 24: Import an Existing ABEL-HDL Source File

As you continue working with the ispLEVER software, you will want to import existing ABEL-HDL source files, in addition to creating new ones. To save time in this tutorial, you will import the remaining ABEL-HDL source file.

To import an existing ABEL-HDL source:

1. In the Project Navigator, choose **Source > Import** to open the Import File dialog box.
2. Select **counter.abl** and click **Open**.

The Project Navigator should now look like the following image. You will take a look at the “insides” of this module in the next lesson.



Lesson 5: Navigating the Design

The Hierarchy Navigator program allows you to navigate through a schematic design that consists of a top-level schematic, lower-level schematics, and HDL modules. The Hierarchy Navigator loads a full hierarchical design all at once so that you can view it in its complete form, rather than as individual sources. Every schematic sheet and behavioral file at all levels of hierarchy is included.

This lesson shows you how to use the Hierarchy Navigator to perform several useful functions.

The following tasks are covered in this lesson:

- ◆ Task 25: Open the Hierarchy Navigator
- ◆ Task 26: Push into the COUNTER Block Symbol

- ◆ Task 27: Access Connectivity Information for the COUNTER Block
- ◆ Task 28: Query a Net

Task 25: Open the Hierarchy Navigator

You can open the Hierarchy Navigator from within the Project Navigator. The Hierarchy Navigator performs several important functions:

- ◆ It verifies the correctness and consistency of a design's wiring. Verification occurs at each level in the design, and across all the levels, from top to bottom.
- ◆ It provides the environment in which you can analyze and optimize the circuit's performance.
- ◆ It prepares the design data for later steps in the design process, for example, creating netlists.

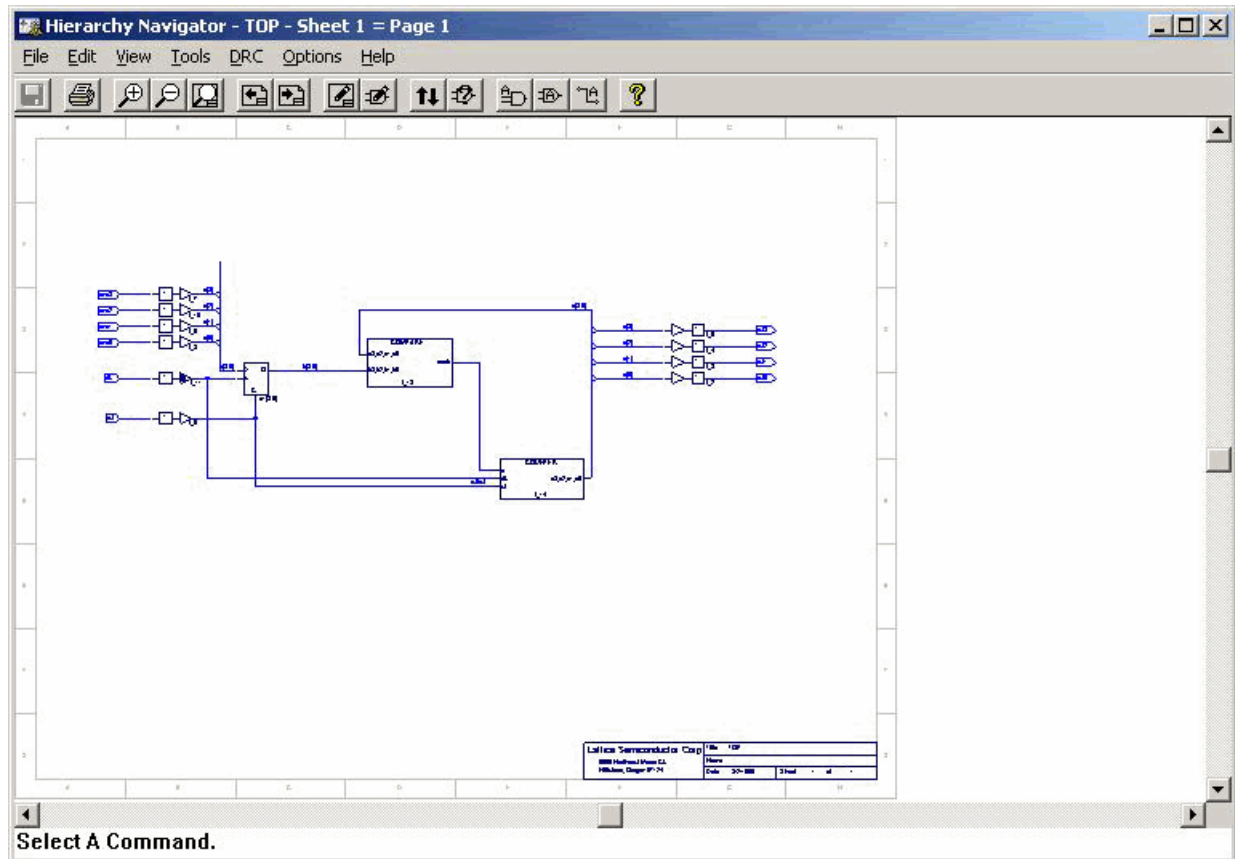
To open the Hierarchy Navigator:

1. In the Sources in Project window, select the **top** schematic source (top.sch).

Notice that the Navigate Hierarchy process is displayed at the top of the Processes for Current Source window. Also notice that Navigate Hierarchy is visible only when a schematic source is selected.

2. In the Processes for Current Source window, double-click the **Navigate Hierarchy** icon.

The Hierarchy Navigator opens with sheet 1 of the selected schematic source loaded.



Note

Remember, this is not the Schematic Editor. You cannot edit the schematic or one of its symbols in the Hierarchy Navigator. However, you can open editors from the Hierarchy Navigator to make changes to a specific schematic element.

Task 26: Push into the Counter Block Symbol

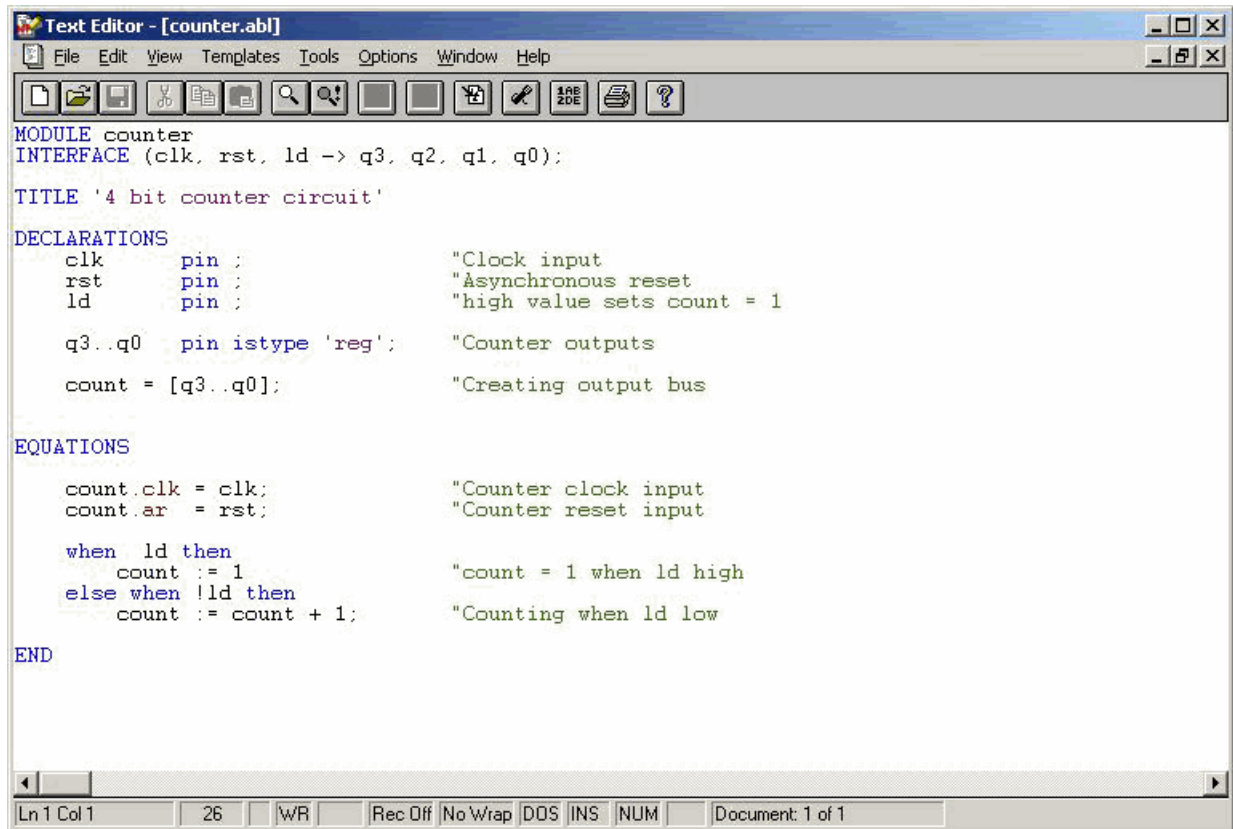
You can use the Push/Pop command on the View menu to move down and up (respectively) through the hierarchical levels of a design. This command works on both schematics and ABEL-HDL modules.

You may want to use the Zoom In command to view the COUNTER module before starting this step. Right-click to cancel the zoom command.

To push into the COUNTER block symbol:

1. Choose **View > Push/Pop**. The cursor changes to a crosshair.
2. To move down a level into the COUNTER module, click inside the **COUNTER** schematic symbol.

The Text Editor opens with the ABEL-HDL description of the module.



```

Text Editor - [counter.abl]
File Edit View Templates Tools Options Window Help
MODULE counter
INTERFACE (clk, rst, ld -> q3, q2, q1, q0);
TITLE '4 bit counter circuit'
DECLARATIONS
  clk      pin ;           "Clock input
  rst      pin ;           "Asynchronous reset
  ld       pin ;           "high value sets count = 1
  q3..q0   pin istype 'reg'; "Counter outputs
  count = [q3..q0];        "Creating output bus
EQUATIONS
  count.clk = clk;         "Counter clock input
  count.ar  = rst;         "Counter reset input
  when ld then
    count := 1;            "count = 1 when ld high
  else when !ld then
    count := count + 1;    "Counting when ld low
END
Ln 1 Col 1 | 26 | WR | Rec Off | No Wrap | DOS | INS | NUM | Document: 1 of 1

```

- View the contents of the COUNTER module. When you finish, close the Text Editor, keeping the Hierarchy Navigator open.

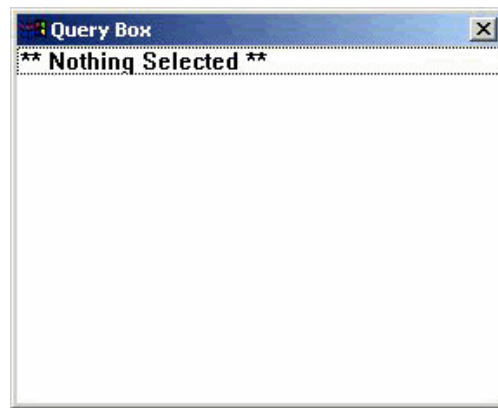
Task 27: Access Connectivity Information for the Counter Block

You can use the Query command to display additional information about circuit elements. The information appears in a text box that pops up when the first element is selected. The box is updated when another element is selected.

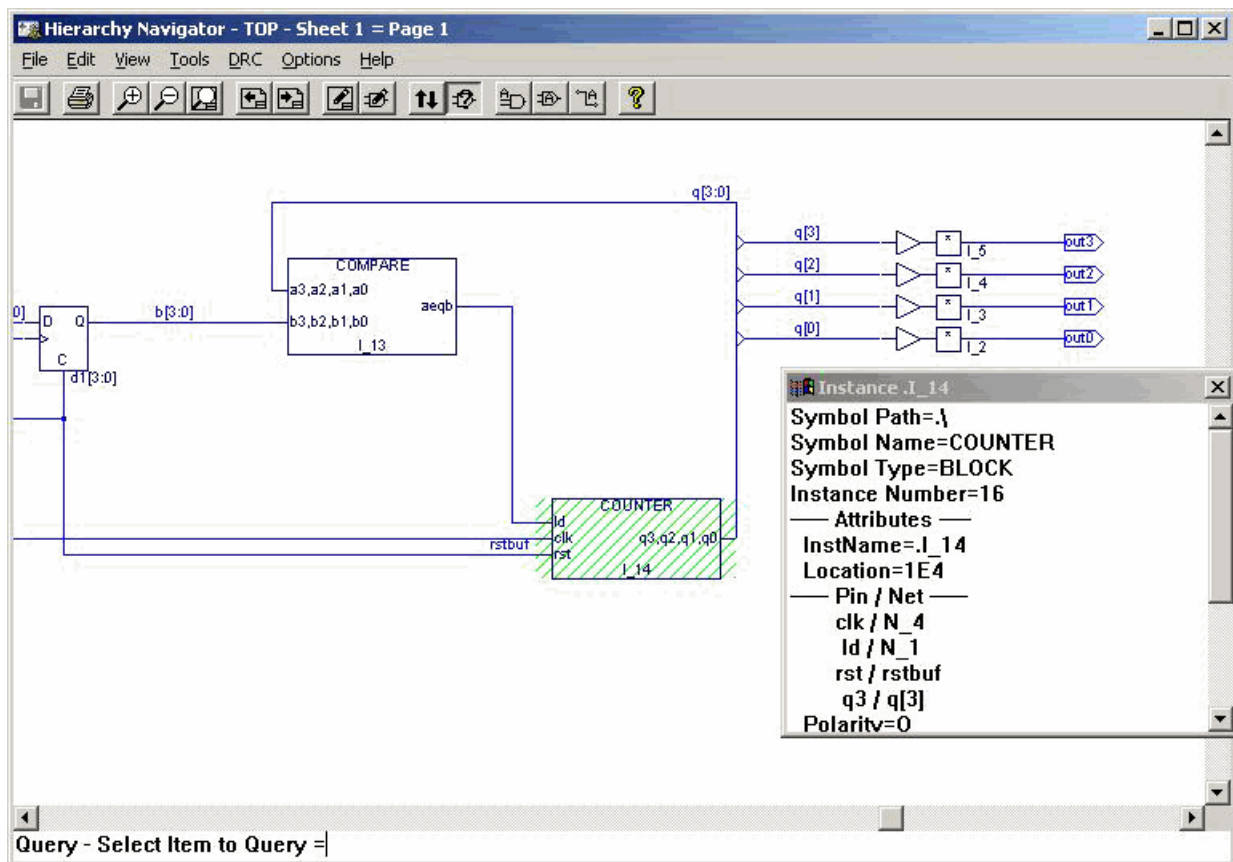
In this step, you will use the Query command to query the Counter ABEL-HDL module for information.

To query the COUNTER block symbol:

- In the Hierarchy Navigator, choose **DRC > Query**. The Query text box opens with the message `Nothing Selected`. Move the Query Box so that you can see the COUNTER block.



- Click the **COUNTER** block. The Query Box is no longer titled "Query." Instead, it now reflects the name of the instance you have selected and shows various items of information about the COUNTER block.



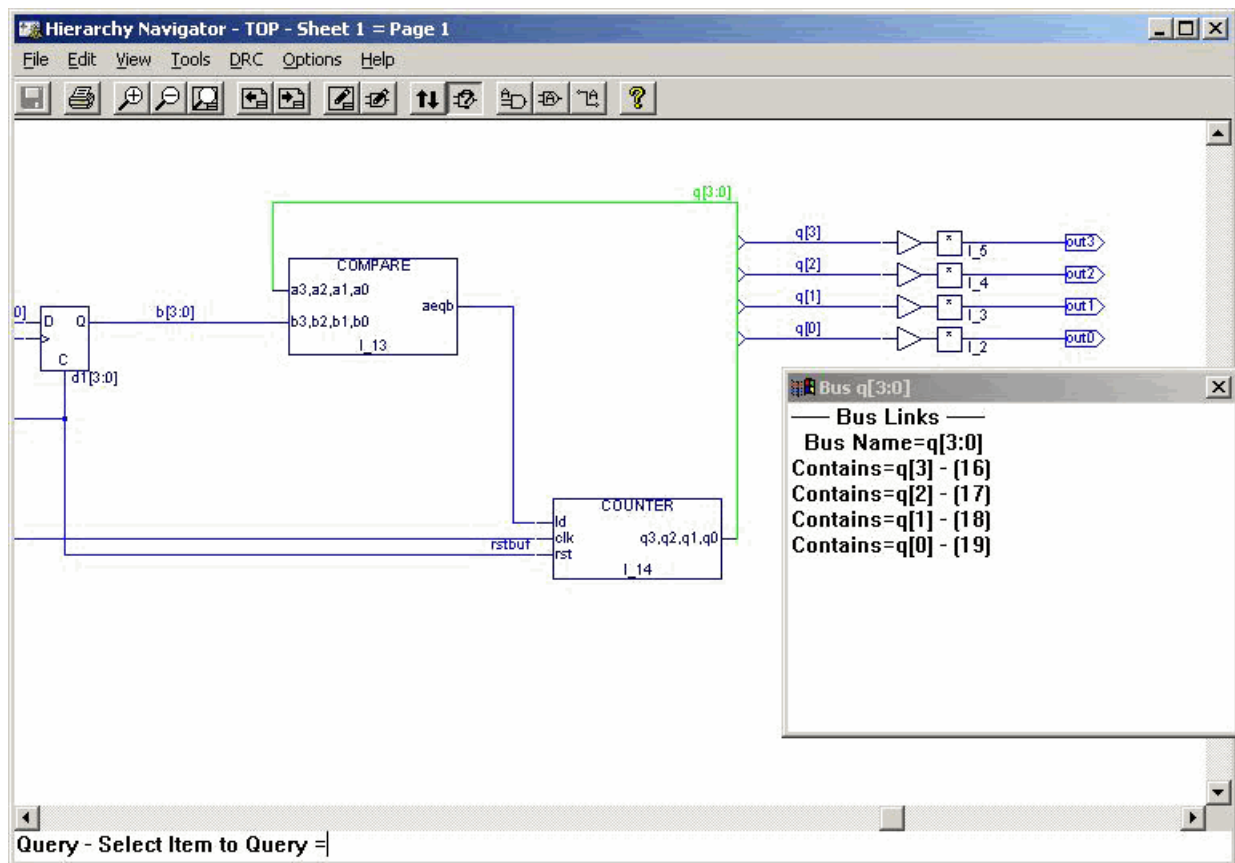
- Look at the **Pin/Net** section. The Hierarchy Navigator knows which nets are connected to which pins, and displays instance names, reference designators, or symbol names that have been assigned to that symbol.
- Keep the Query Box open for the next task.

Task 28: Query a Net

Although design problems are usually observed at the top level, the source of the problems is often at a lower level. Tracing signals from the primary outputs down through the hierarchy can greatly aid debugging. For instance, to determine if a net needs more buffers, use the Query command to determine what components are attached to the net and what will be affected by changes to the net.

To query a net:

1. With the Query command still active, click the bus labeled **q[3:0]** in the schematic. A list of its content signals appears in the Query Box.



Observe that the signals displayed are links (pointers) to the individual nets.

2. In the Hierarchy Navigator, click **q[1]**. The Query Box shows new information for the net. (You will not see information about the bus and net at the same time. When one window opens, the other window closes.)
3. In the Query Box, click one of the connections to a net. The cursor automatically moves to the component where the connecting pin is located. The cursor automatically moves to any page in the hierarchy, no matter which level, if necessary to display the selected pin.
4. Close the Query Box and the Hierarchy Navigator. If you see the Data has been modified prompt, click **NO**.

Lesson 6: Running Functional Simulation

Functional simulation is the process of simulating the functionality of your RTL design before it is compiled, enabling you to find and correct basic design errors sooner. While functional simulation verifies your Boolean equations, it does not indicate timing problems.

The ispLEVER software supports functional simulation for any Lattice Semiconductor device using the Lattice Logic Simulator or ModelSim™ from Model Technology. These simulators operate in both stand-alone and integrated environments.

This lesson shows you how to use the Lattice Logic Simulator to run integrated functional simulation.

Tasks covered in this lesson are the following:

- ◆ Task 29: Import the Test Vector File
- ◆ Task 30: Run Functional Simulation

Task 29: Import the Test Vector File

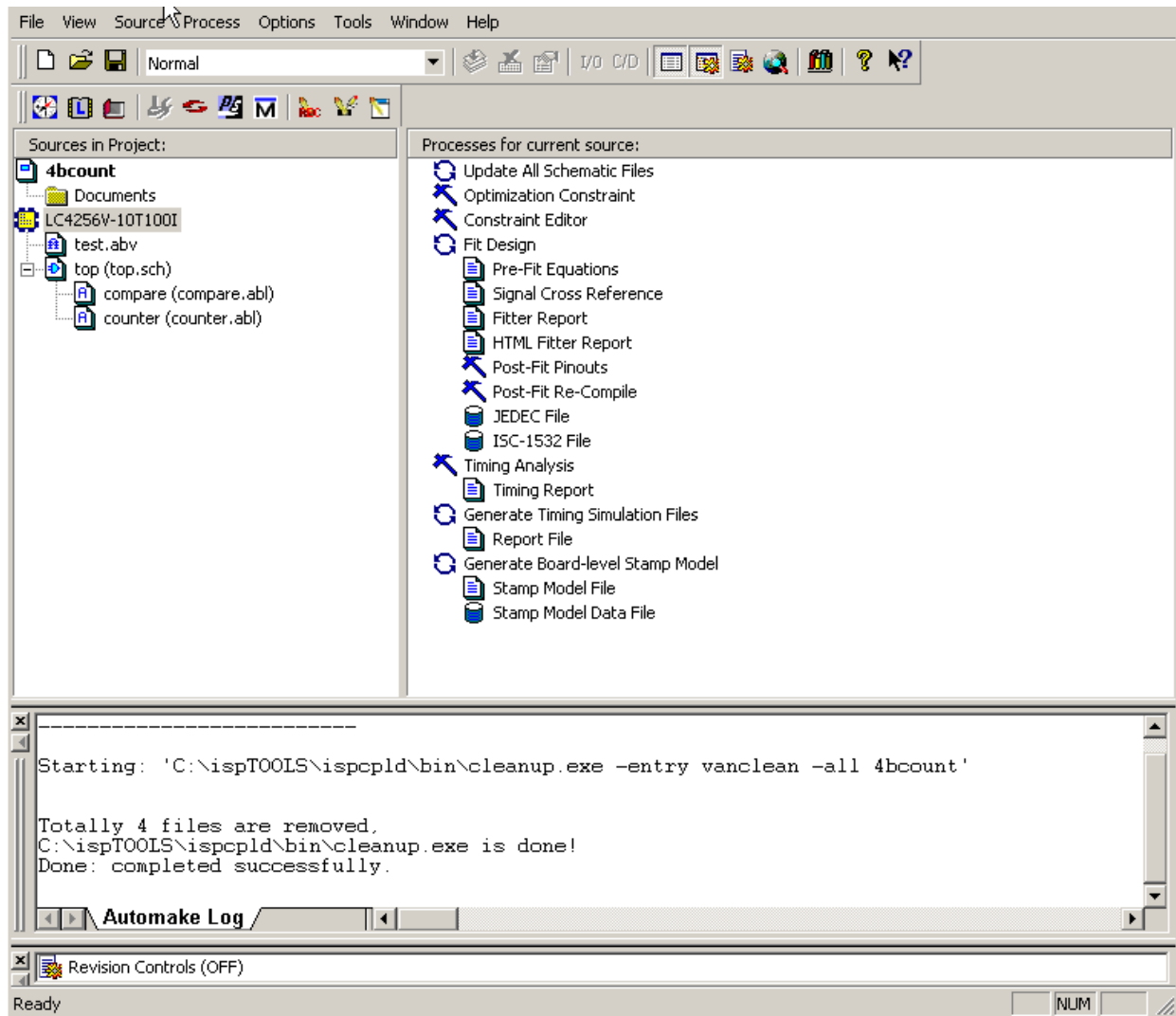
You can create a test vector file in a text editor using proper keywords. Test vectors are sets of input stimulus values and corresponding expected outputs that can be used with both functional and timing simulators. Test vectors can be specified either in a top-level ABEL-HDL source or in a separate ABEL-HDL test vector format file called an `.abv` file. The `.abv` file is considered a text document and is kept above the device level in the Sources in Current Project window. Whether the test vectors are part of a top-level ABEL-HDL source (`.abl`) or are in a separate file, they will be compiled and passed to the simulator.

In this task, you will import a test vector that has already been created for you.

To import a test vector source file:

1. In the Project Navigator, choose **Source > Import** to open the Import File dialog box.
2. Select the `test.abv` file and click **Open**.

The file appears just below the device in the Sources in Project window.



3. View the contents of the test vector file in the Text Editor by double-clicking the file name in the Sources in Project window.

```

module test

clk,rst                pin:
end3,end2,end1,end0   pin:

out3,out2,out1,out0   pin:

endv = [end3,end2,end1,end0];
outv = [out3,out2,out1,out0];

test_vectors

([clk,rst,endv]        ->      [outv]);

[.x.,1,^h5]           ->      [^h3];
[.c.,0,^h5]           ->      [^h1];
[.c.,0,^h5]           ->      [^h2];
[.c.,0,^h5]           ->      [^h3];
[.c.,0,^h5]           ->      [^h4];
[.c.,0,^h5]           ->      [^h5];
[.c.,0,^h5]           ->      [^h1];
[.c.,0,^h5]           ->      [^h2];

END

```

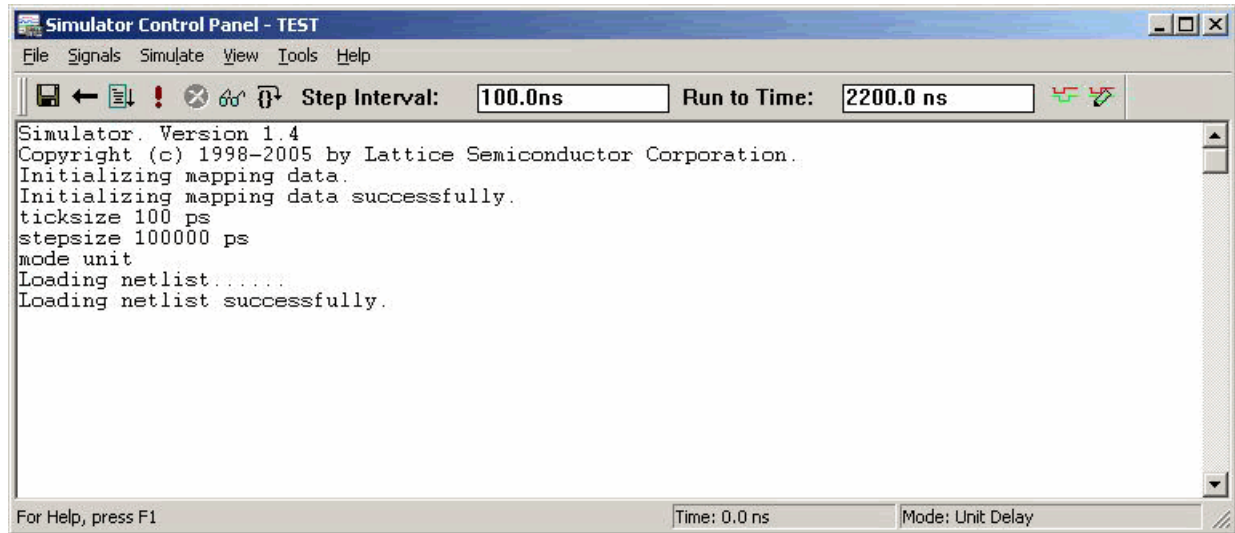
4. Close the Text Editor.

Task 30: Run Functional Simulation

After creating a test vector file, you can verify the behavior of your design by performing functional simulation. You can run integrated functional simulation using the Functional Simulation process in the design flow. This process opens the Lattice Logic Simulator and loads the functional netlist.

To perform functional simulation:

1. In the Project Navigator Sources in Project window, select the ABEL-HDL test vector file `test.abv`.
2. In the Processes for Current Source window, double-click the **Functional Simulation** process to launch the Lattice Logic Simulator Control Panel and load the netlist.

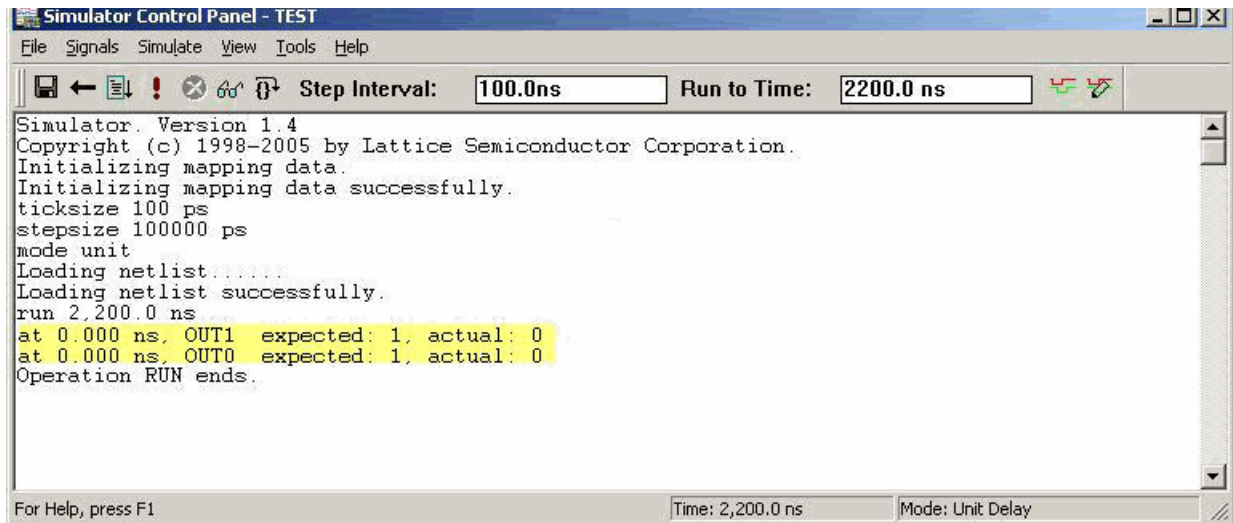


3. In the Simulator Control Panel, choose **Simulate > Run** to start the simulation.

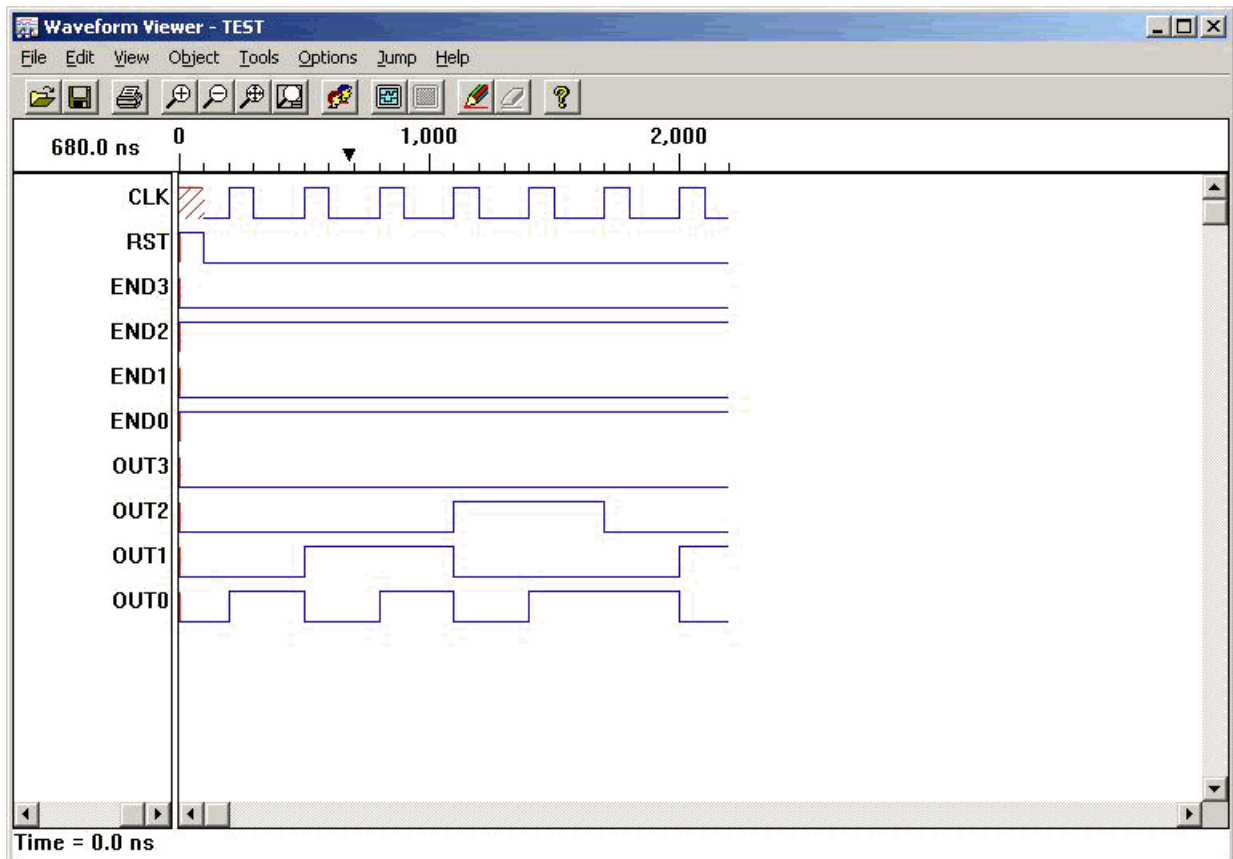


Note

If the simulation contains errors, the control panel displays a message similar to the following highlighted example.



After functional simulation is complete, the simulator automatically opens the Waveform Viewer, which is the primary tool for viewing simulation results. The Viewer graphically depicts the activity on any node in the simulation database. The Viewer automatically updates as simulation progresses.



4. Close the Waveform Viewer and the Lattice Logic Simulator Control Panel.

5. In the Project Navigator, choose **File > Save** to save the design before going to the next module.

Module 2: Design Implementation

For ispLEVER CPLD designs, the design implementation process consists of several steps, which generally include applying user constraints, compiling, optimizing, and fitting the design. These individual steps happen automatically and in sequence when you run the Fit Design process in the Project Navigator.

This module shows you how to set constraints, and then leads you through the processes of compiling, optimizing, and fitting your CPLD design. Also, you will learn how to set report viewing options to view the various report files generated by the ispLEVER software.

Prerequisites

You should complete Module 1 before proceeding with this module.

Learning Objectives

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

- ◆ Use the Constraint Editor to pre-assign pin locations
- ◆ Use the Constraint Editor Package View window to graphically view the actual pin assignments in the target device, and assign pin locations using the drag-and-drop technique.
- ◆ Run the ispLEVER fitter
- ◆ Set fitter report viewing options and read the fitter report

Time to Complete This Module

The time to complete this module is about 20 minutes.

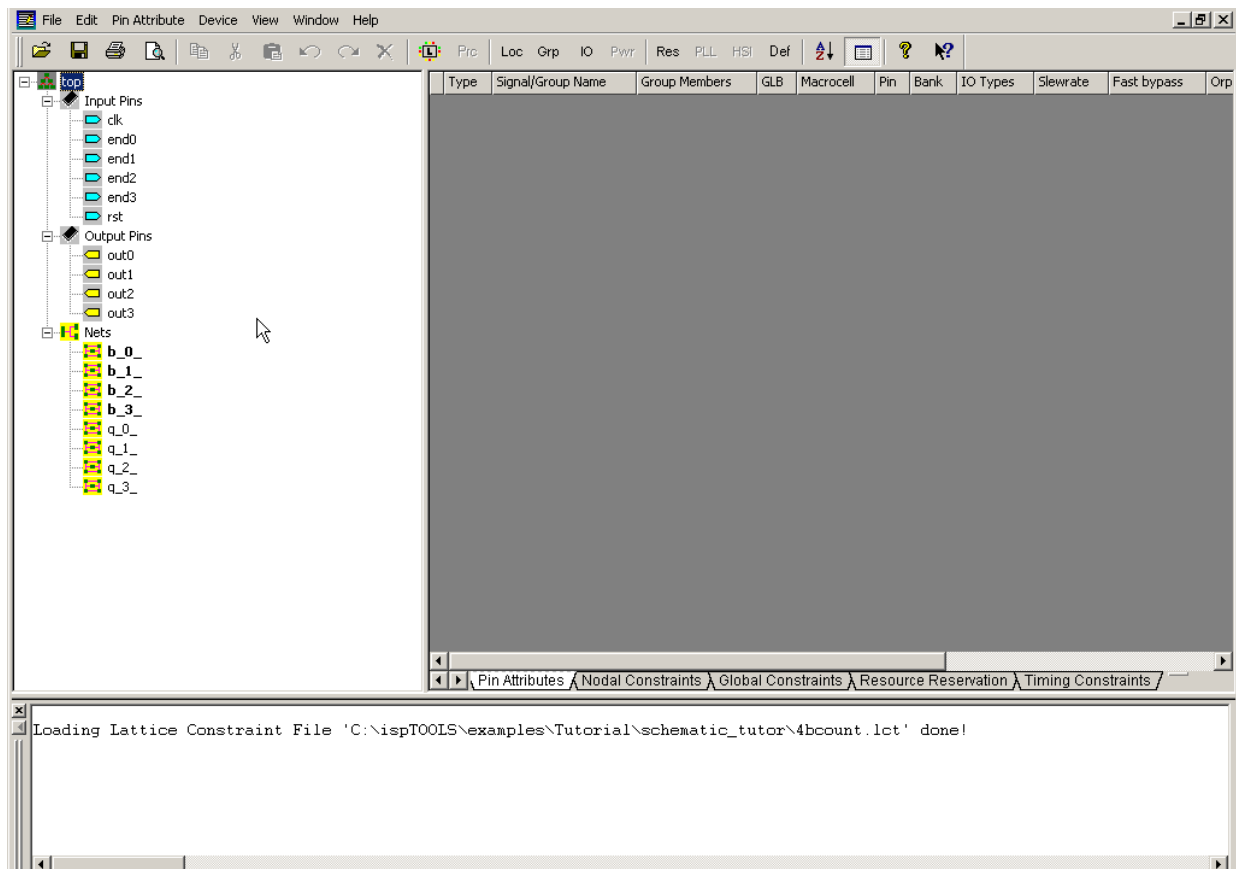
Task 1: Pre-Assign Pin Locations

In some cases, you may want to assign constraints before running the fitter. For example, you may want to pre-assign pin and node locations. The Constraint Editor enables you to specify various constraints using the graphical interface such as pin and node assignments, group assignments, pin reservations, power level settings, output slew rates, and JEDEC file options. The Constraint Editor reads the constraint file and displays the constraint settings in the main window. You can modify the constraint file using the function dialog boxes available from the toolbar. However, some constraints can be modified directly in the main window.

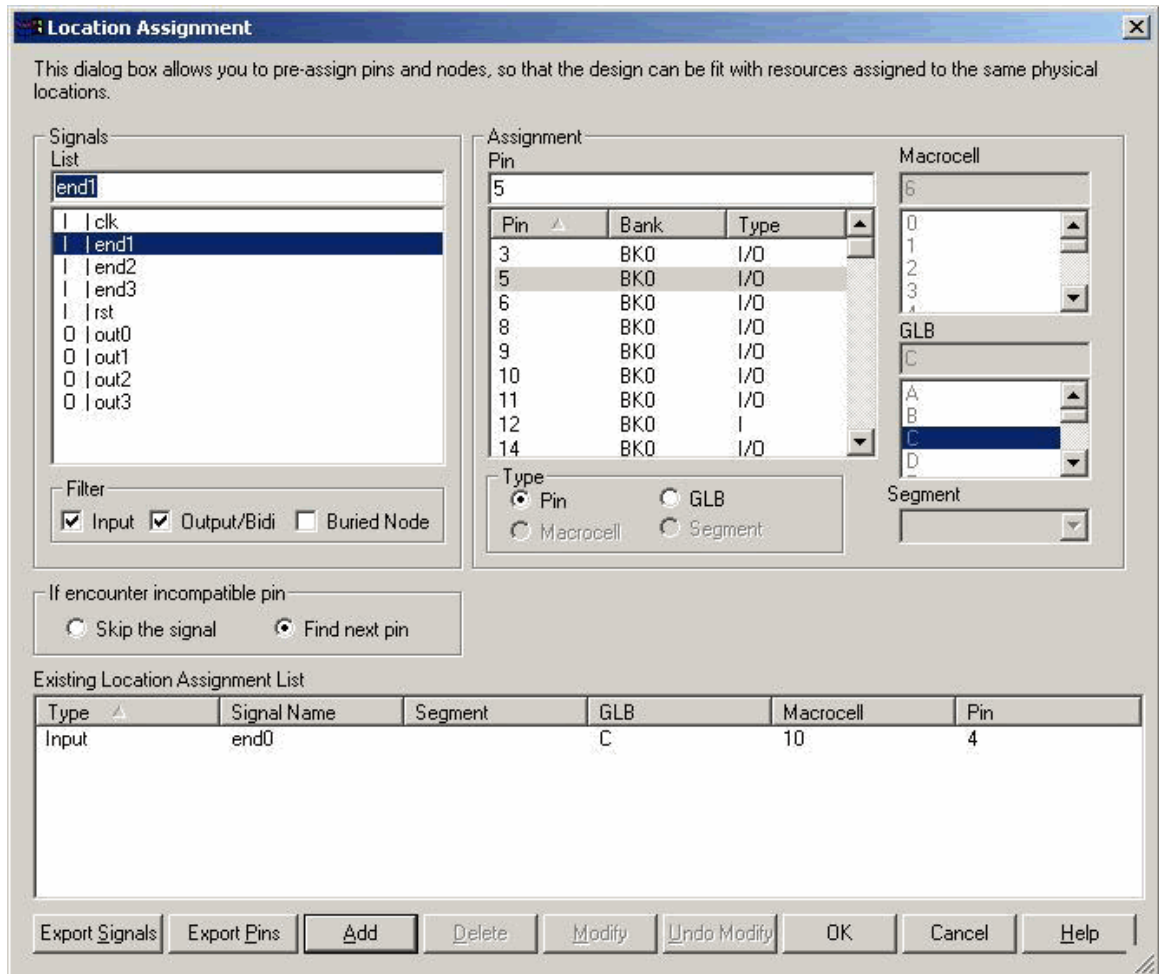
The Constraint Editor implements simple error checking to ensure that the user assignments or constraints are applicable to the selected device and that there are no conflicting assignments. If the user constraints do not apply to the selected device, or are conflicting with the selected device, the Constraint Editor displays these constraints in red.

To assign constraints to the design:

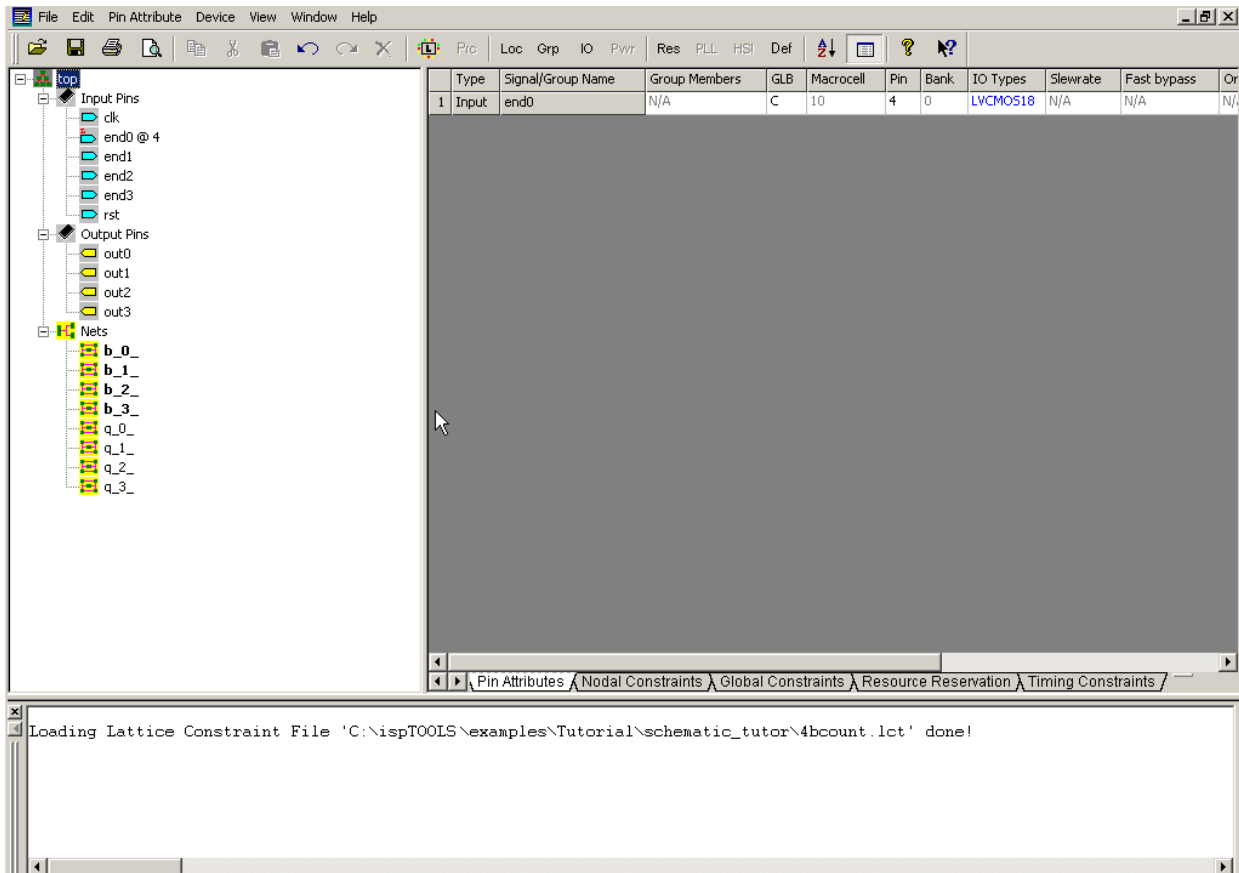
1. In the Sources in Project window, select the target device. In the Processes for Current Source window, double-click **Constraint Editor** to open it. Expand the tree in the left pane.



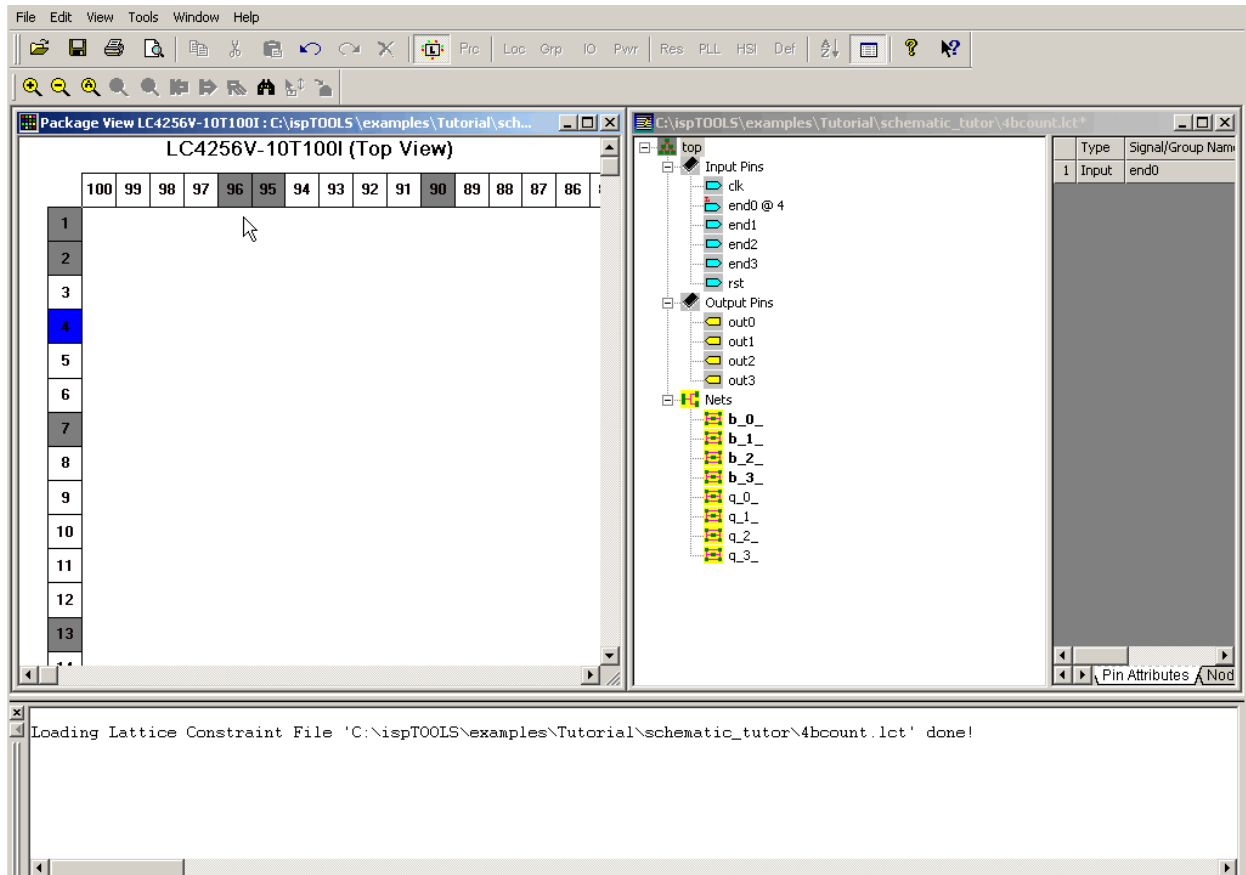
2. Choose **Pin Attribute > Location Assignment** to open the dialog box and do the following:
 - a. Under Signals List, select **end0**.
 - b. Under Pin Assignment, select **4**.
 - c. Click **Add**.



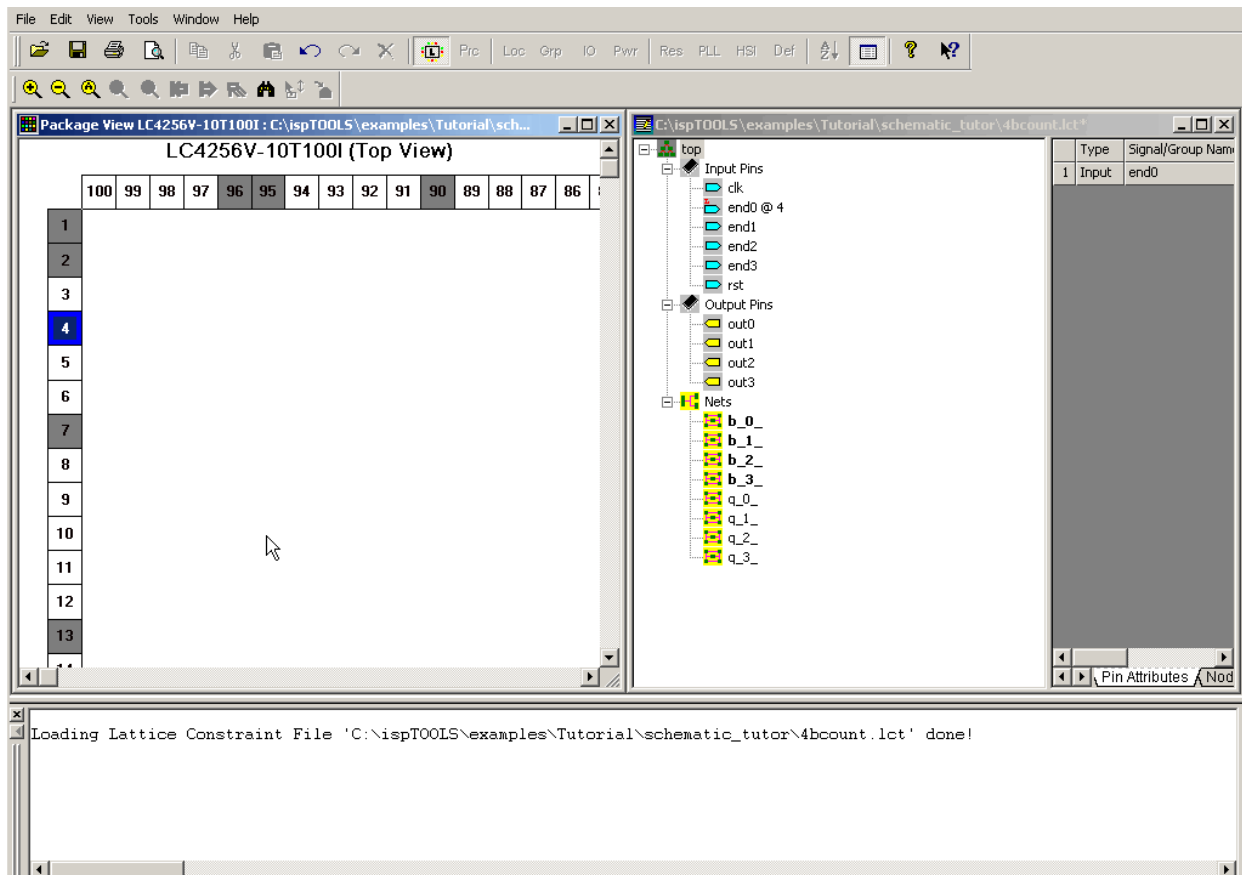
3. Click **OK** to close the dialog box.
4. In the Constraint Editor, notice the pin location information in the Pin Attributes sheet. In the Signal List Pane on the left side of the main window, click the plus sign [+] in front of **Input Pins** to expand the tree view. Notice that the end0 pin has a “locked” icon next to it, indicating that the pin location has been assigned.



5. You can also use the Constraint Editor Package View window to graphically view the actual pin assignments in the target device, and assign pin locations using the drag-and-drop technique. By default, the pins are highlighted in the following colors:
 - ◆ Gray: the system (non-user)
 - ◆ Lime: reserved pins
 - ◆ Blue: assigned input pins
 - ◆ Yellow: output pins
 - ◆ Magenta: bidirectional pins
 - ◆ Blank: unused pins
6. Choose **Device > Package View** to open the Package View window.

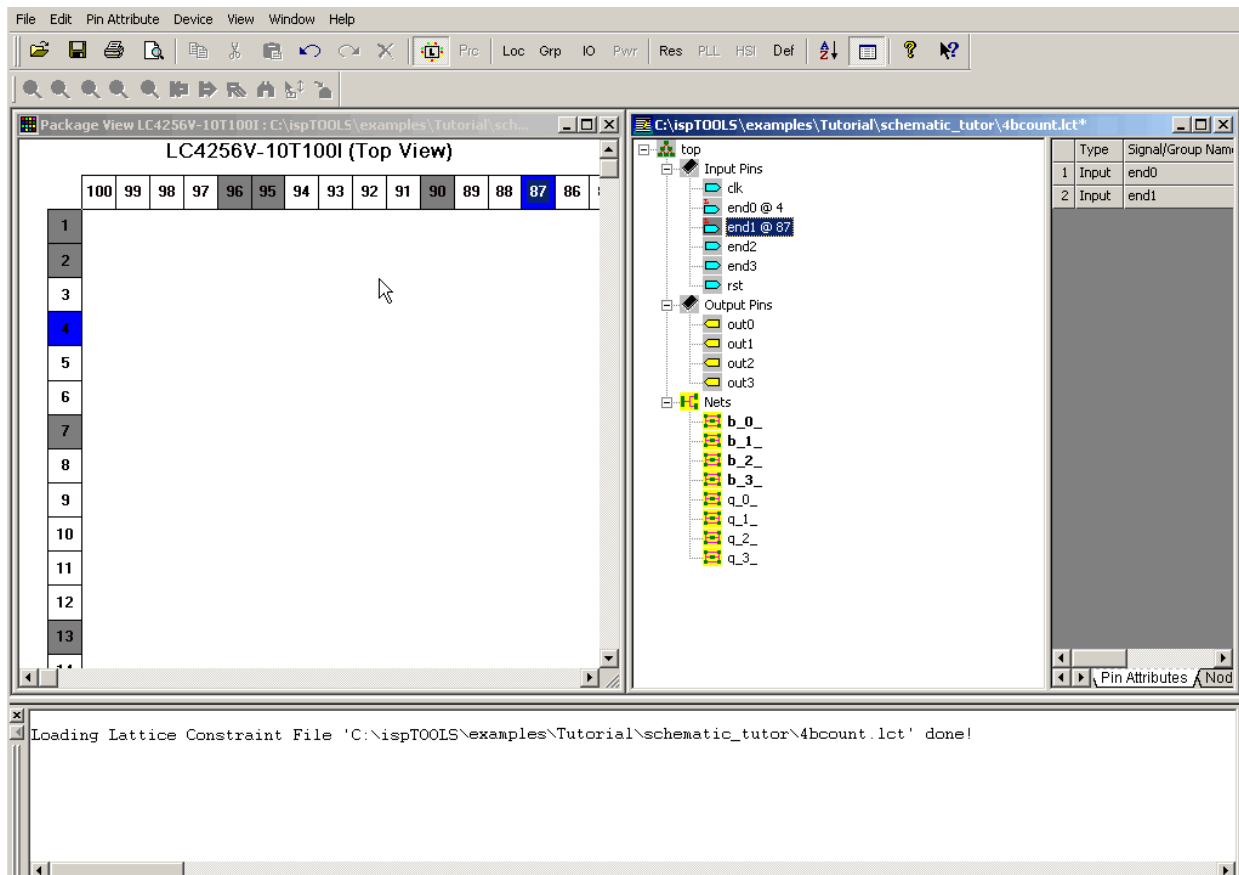


- To easily locate a previously assigned pin, click the Find icon (🏠) on the toolbar. In the Find dialog box, type **4** in the Name box, select **Device Pin**, and then click **Find**.



The view shifts to the specified location.

- You can also assign pin locations by using the Package View window. If the Package View and the Signals List are not already side by side, choose **Window > Tile Vertically**. Make sure the Package View window is showing the 4 pin and the signal list is expanded to show the input pins.
- Select **end1** from the Signal List and drag it to the **87** location in the Package View window. Notice the 87 location is blue (assigned input pins), the "locked" icon appears next to end1 in the Signals List, and end1 has been added to the Pin Attributes sheet.



10. Close the Package View window. Choose **File > Save** to save the pin assignment. Close the Constraint Editor.

Task 2: Fit the Design

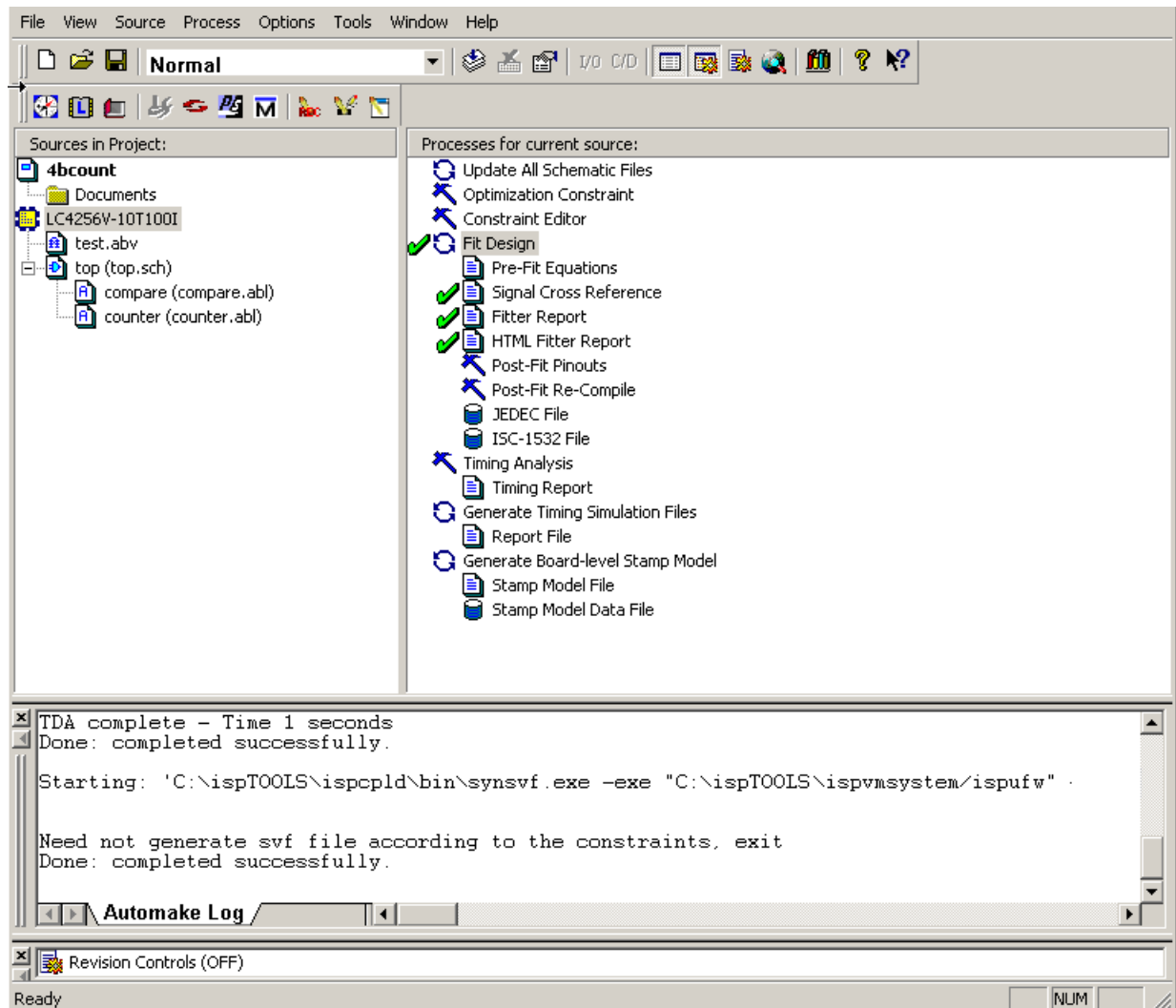
The ispLEVER software has a single user interface with all options preset to deliver the highest possible push-button performance. At the end of a successful fitter run, the ispLEVER software generates a JEDEC file, as well as a fitter report, so that you can see how the ispLEVER software has routed the design and utilized resources on the part.

Fitting is an integrated process that includes these steps:

- ◆ **Compiling** - Changes your design entry format into Boolean equations, which serve as input to simulation and device implementation programs.
- ◆ **Optimizing** - Runs a set of options that let you achieve the highest possible performance in the smallest possible device, for most designs.
- ◆ **Partitioning** – Partitions the design after optimization into individual blocks on the specified device. Partitioning assigns logic to specific blocks on the basis of several considerations.
- ◆ **Fitting** - Runs the device fitter. A fitter report is generated whether or not the process is successful. The fitter also generates the JEDEC file.

To fit a design:

1. In the Sources in Project window, select the target device.
2. In the Processes for Current Source window, double-click **Fit Design**. A message appears in the Output Panel telling you that the fit process completed successfully. Also, a green check mark appears to the left of the Fit Design process.

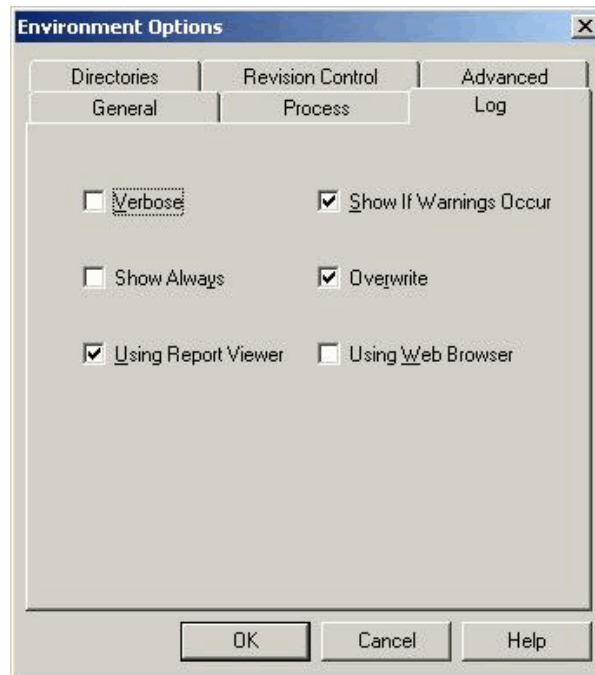


Task 3: Set Report Viewing Options

By default, the ispLEVER software opens report files in the Project Navigator Output Panel. However, some reports may be too large to allow easy viewing in the Output Panel.

To open a report in the Report Viewer:

- ◆ Choose **Options > Environment** to open the dialog box. Then click the **Log** tab and select **Using Report Viewer**. Click **OK** to close the dialog box.



Task 4: Read the Fitter Report

The Fitter Report displays statistics and information, such as utilization numbers and pin assignments, on the fitting process of your design.

To view the Fitter Report:

1. In the Processes for Current Source window, double-click the **Fitter Report** process to open the Fitter Report in the Report Viewer. View the report and then close the Report Viewer.

The screenshot shows a window titled "Report Viewer - [4bcount.rpt]". The window contains a text-based report with the following content:

```

- ispLEVER_CP_ALL Fitter Report File -
- Copyright(C), 1992-2005, Lattice Semiconductor Corporation -
- All Rights Reserved. -

The Basic/Detailed Report Format can be selected in the dialog box
Tools->Fitter Report File Format...

Project_Summary
~~~~~
Project Name      : 4bcount
Project Path     : C:\ispTOOLS\examples\Tutorial\schematic_tutor
Project Fitted on : Thu Jul 07 11:42:04

Device          : M4256_64
Package         : 100
GLB Input Mux Size : 33
Available Blocks : 16
Speed           : -10
Part Number     : IC4256V-10T100I
Source Format    : ABEL_Schematic

// Project '4bcount' Fit Successfully! //

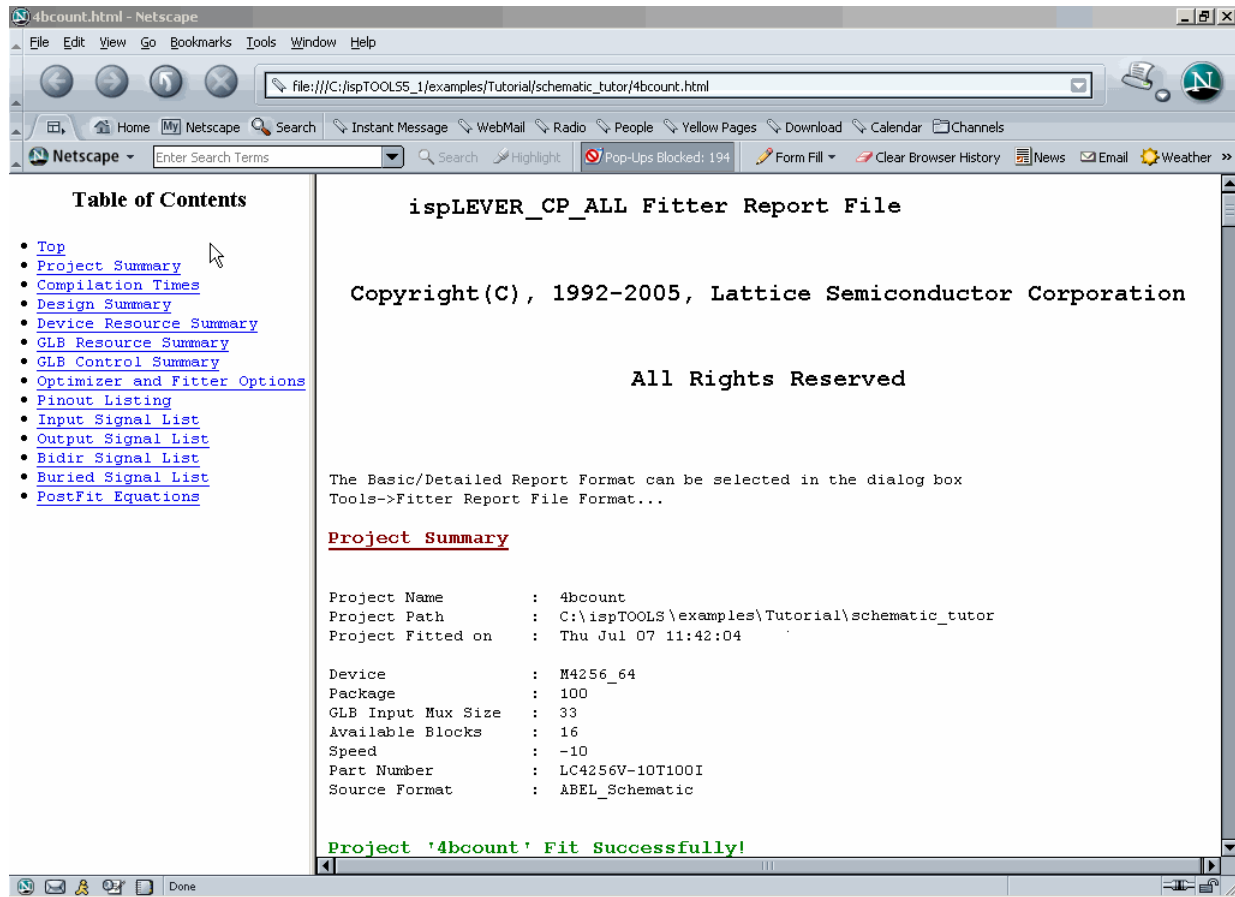
Compilation_Times

```

At the bottom of the window, there is a status bar with the following information: Ln 1 Col 1, 525, RD, Rec Off, No Wrap, DOS, INS, NUM, Document: 1 of 1.

As you probably noticed, the Fitter Report is divided into several sections and can be quite long, so it is difficult to find the information that you are looking for. As an alternative, you can view the Fitter Report using your HTML browser. This method offers the benefit of easy browsing using a navigation panel with links to each major section in the report.

2. In the Processes for Current Source window, double-click the **HTML Fitter Report** process to open the report in your browser.
3. Click the **Pinout Listing** navigation link. Find pin number **4**. Look all the way to the right to find the **end0** signal, which is the pin location constraint that you set in a previous task in this module.



4. When you are finished looking at the fitter report, close the browser.
5. Choose **File > Save** to save the design.

Module 3: Design Verification

This module provides an overview of the features and operation of the ispLEVER software, focusing on the tasks and tools needed to verify an ispMACH device design.

Prerequisites

You should complete Modules 1 and 2 before proceeding with this module.

Learning Objectives

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

- ◆ Perform timing analysis and simulation and analyze the results
- ◆ Correlate simulation results by using cross-probing
- ◆ Export the netlist and delays for timing simulation
- ◆ Build board-level stamp models of a design

Time to Complete This Module

The time to complete this module is about 20 minutes.

Task 1: Perform Static Timing Analysis

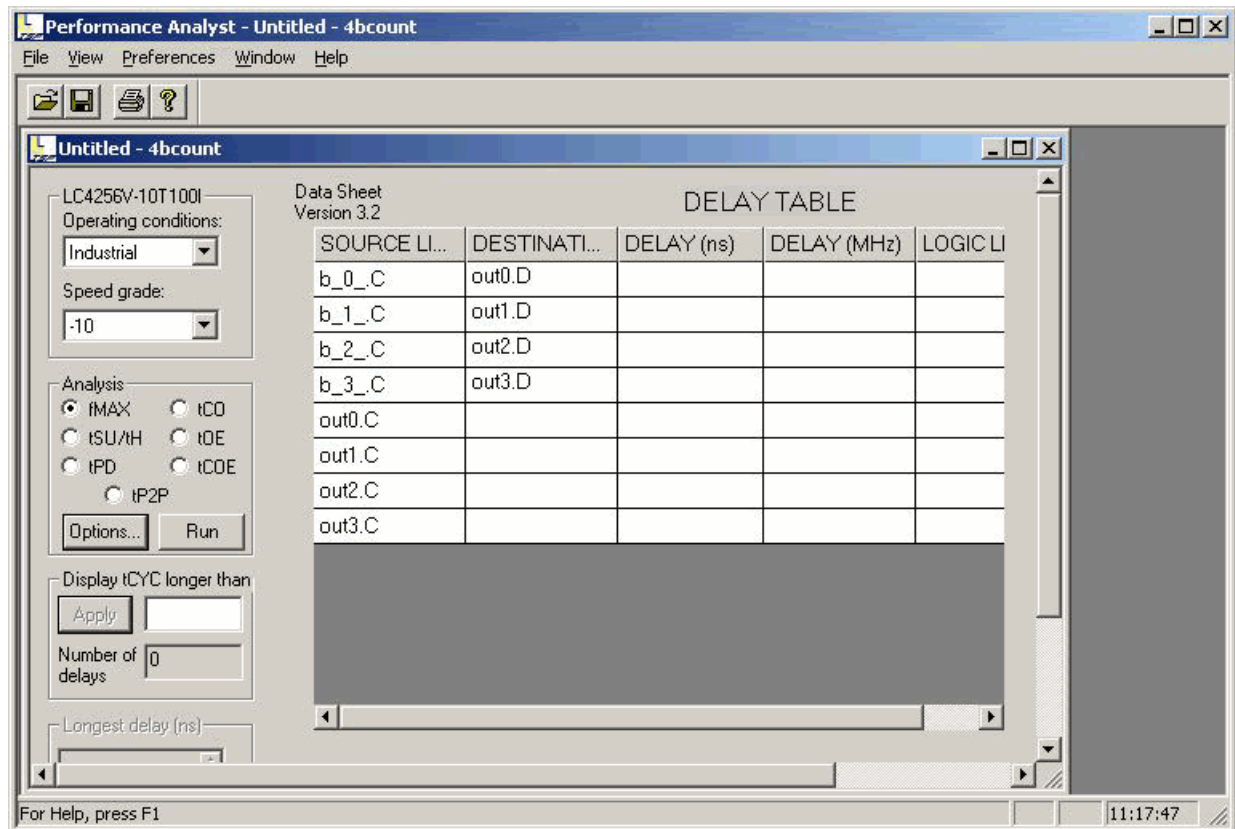
Static timing analysis is the process of verifying circuit timing by totaling the propagation delays along paths between clocked or combinational elements in a circuit. The analysis can determine and report timing data such as the critical path, setup and hold time requirements, and the maximum frequency.

The Performance Analyst traces each logical path in the design and calculates the path delays using the device's timing model and worst-case AC specifications supplied in the device data sheet.

The timing analysis results are displayed in a graphical spreadsheet with source signals displayed on the vertical axis and destination signals displayed on the horizontal axis. The worst-case delay value is displayed in a spreadsheet cell if there is at least one delay path between the source and destination. To more easily identify performance bottlenecks, you can double-click a cell to view the path delay details.

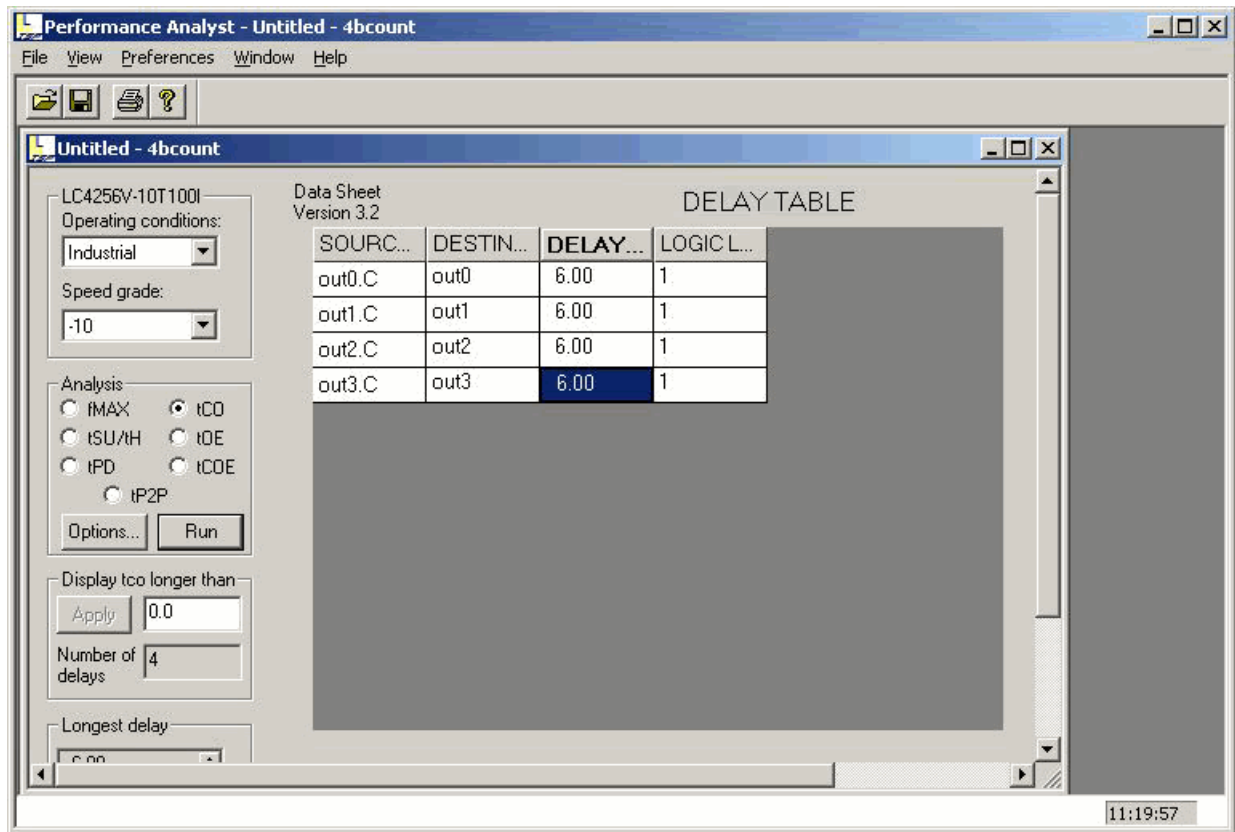
To perform timing analysis:

1. In the Project Navigator Sources window, select the target device. In the Processes for Current Source window, double-click the **Timing Analysis** process to open the Performance Analyst.



The Performance Analyst performs seven distinct analysis types: fMAX, tSU/tH, tPD, tCO, tOE, tCOE, and tP2P. The first type, fMAX, is an internal register-to-register delay analysis. It measures the maximum clock operating frequency, limited by worst-case register-to-register delay. The tP2P type is the path between any two user-specified pins. The remaining five types are external pin-to-pin delay analysis. Timing threshold filters, source and destination filters, and path filters can be used to independently fine-tune each analysis.

- Under Analysis, select **tCO** and then click **Run**. The tCO path trace analysis reports clock-to-out delay starting from the primary input, going through the clock of flip-flops or gate of latches, and ending at the primary output. In this case, it is 6.00 ns.



The lower-left corner shows the longest delay (6.00 ns). Also, in the spreadsheet window, you can see the delays in a source-destination matrix.

- In the spreadsheet window, double-click inside the selected (blue) cell to open the Expanded Path dialog box. This dialog lets you analyze individual timing components used to calculate the timing path. There is a source pin (From) and a destination pin (To). Also shown are the delay type, the delay of that path (Value ns), and the cumulative delay of all the signals.

Expanded Path

Source:out3.C Destination:out3

Ok
Print
Help
Export
Equations

	From	Loc	To	Loc	Delay Type	Value (ns)	Total (ns)
delay path	clk	p89	out3.C	B3	tGCLK_IN+tIC	3.28	3.28
	out3.C	B3	out3.Q	B3	tCOi	1.17	4.45
	out3.Q	B3	out3	p97	tORP+tBUF+t	1.55	6.00

- Click **Equations** to open the Equations dialog box, which shows the functional relationship between the selected source/destination.

Equations

Source: out3.C Destination: out3

```
out3.C = clk; (1 pterm, 1 signal)
```

Help

- Close the Performance Analyst without saving.

Task 2: Run Timing Simulation

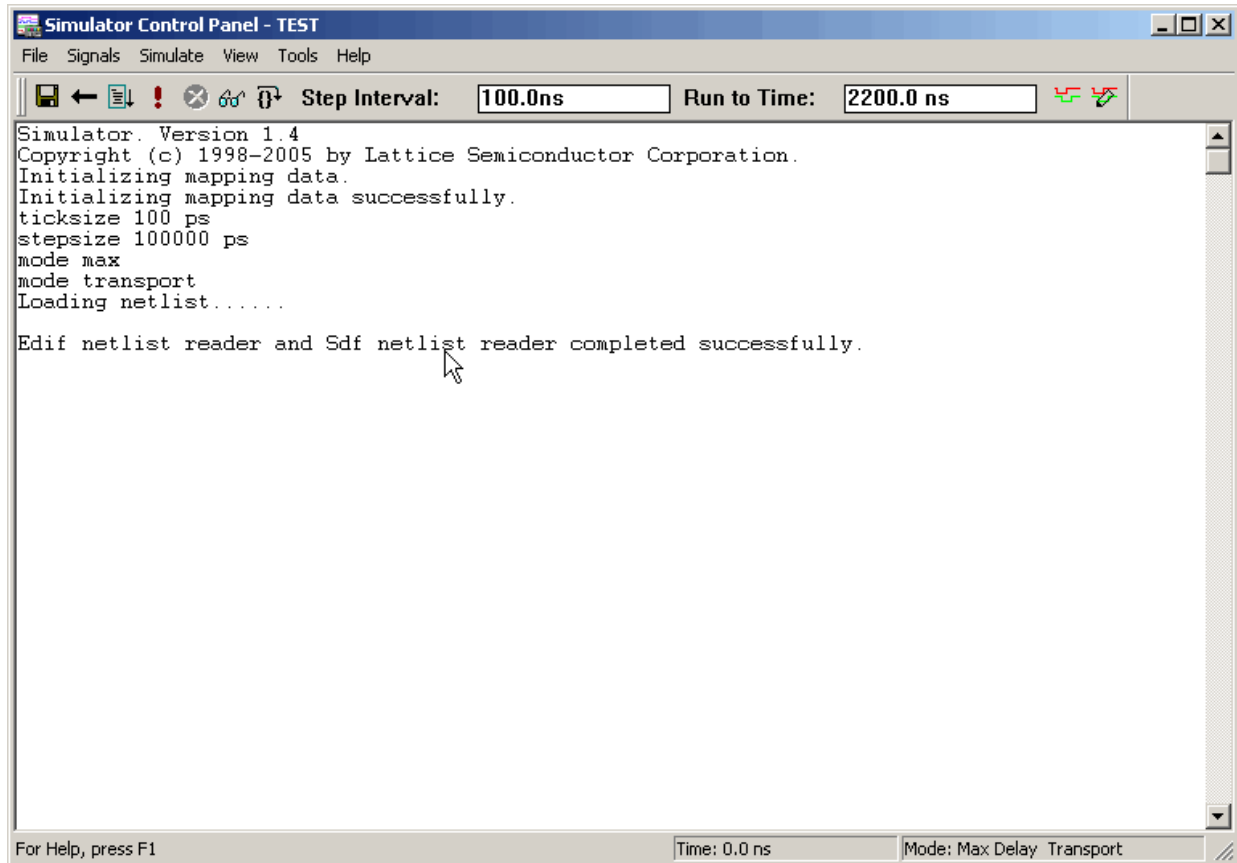
In a previous module, you used the Lattice Logic Simulator for functional simulation. Now you will use the simulator for timing simulation.

Timing simulation differs from timing analysis in a couple of ways. For example, you need a test vector file for simulation. And although both tools provide timing information based on an implemented design, only the Lattice Logic Simulator simulates the design logic as well. Also, the simulator provides graphical waveform display and debugging capabilities. In the end, which tool you use will depend on your specific requirements. You can use the

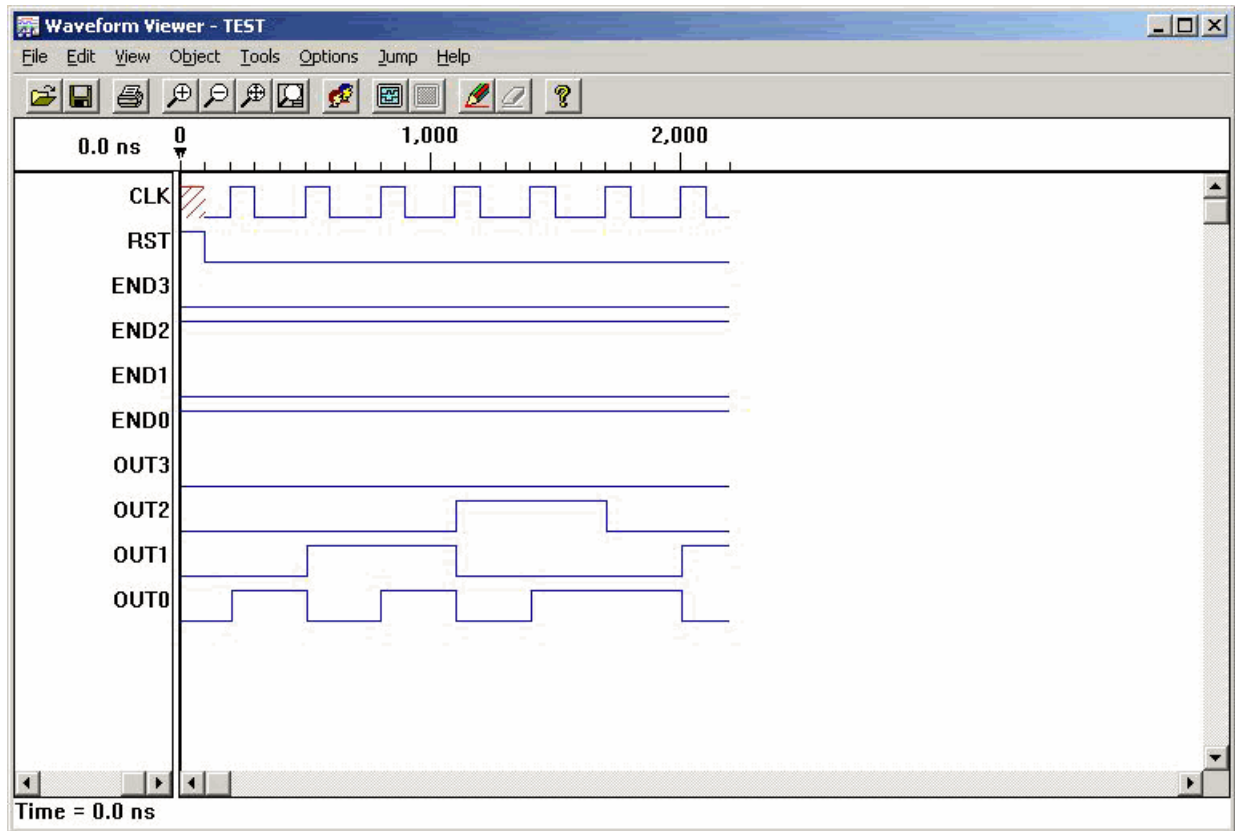
Performance Analyst for a quick critical path analysis, and use the Lattice Logic Simulator for a more detailed and thorough simulation and analysis.

To perform timing simulation:

1. In the Project Navigator Sources window, select the `test.abv` test vector file.
2. In the Processes for Current Source window, double-click the **Timing Simulation** process to open the Lattice Logic Simulator Control Panel. In the toolbar, make sure that the Step Interval is set to **100.0 ns** and Run to Time is set to **2200.0 ns**.




3. Choose **Simulate > Run** to start the timing simulation and open the Waveform Viewer.

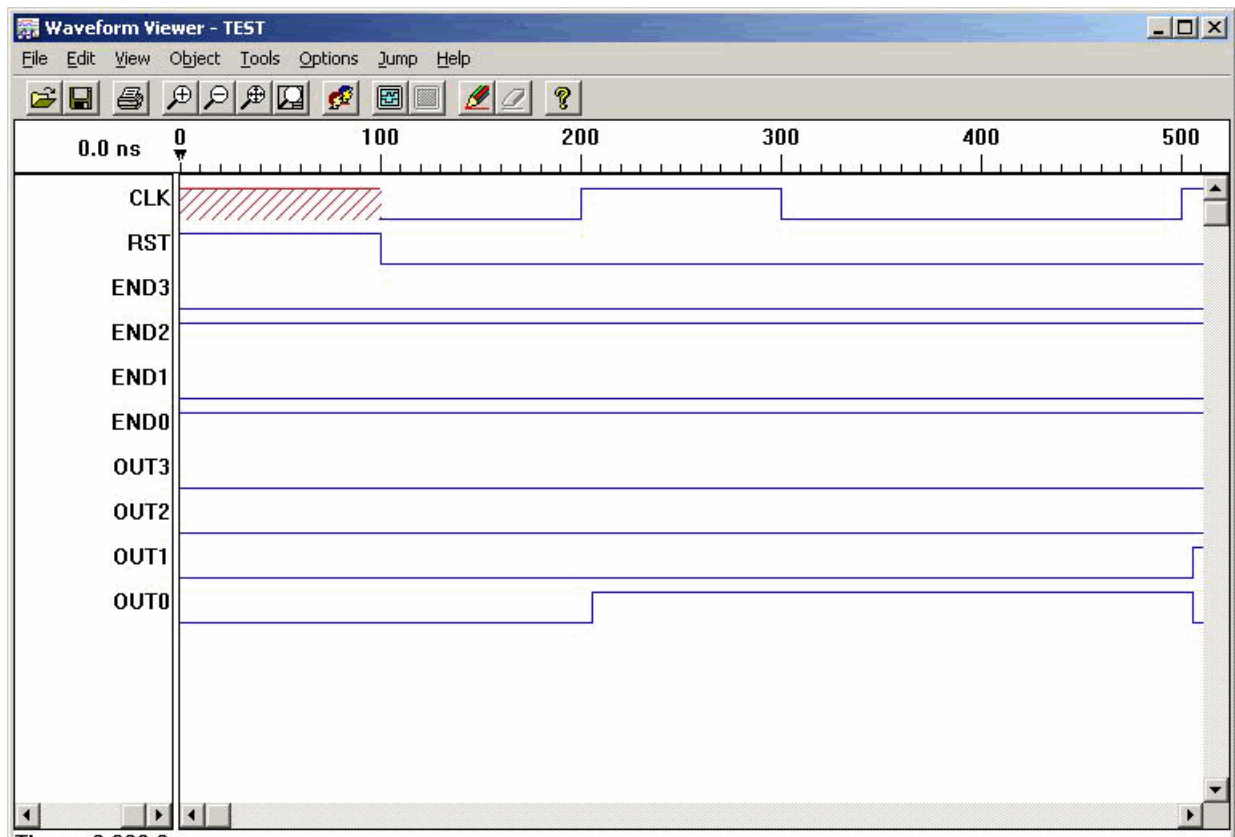


Task 3: Analyze the Simulation Results

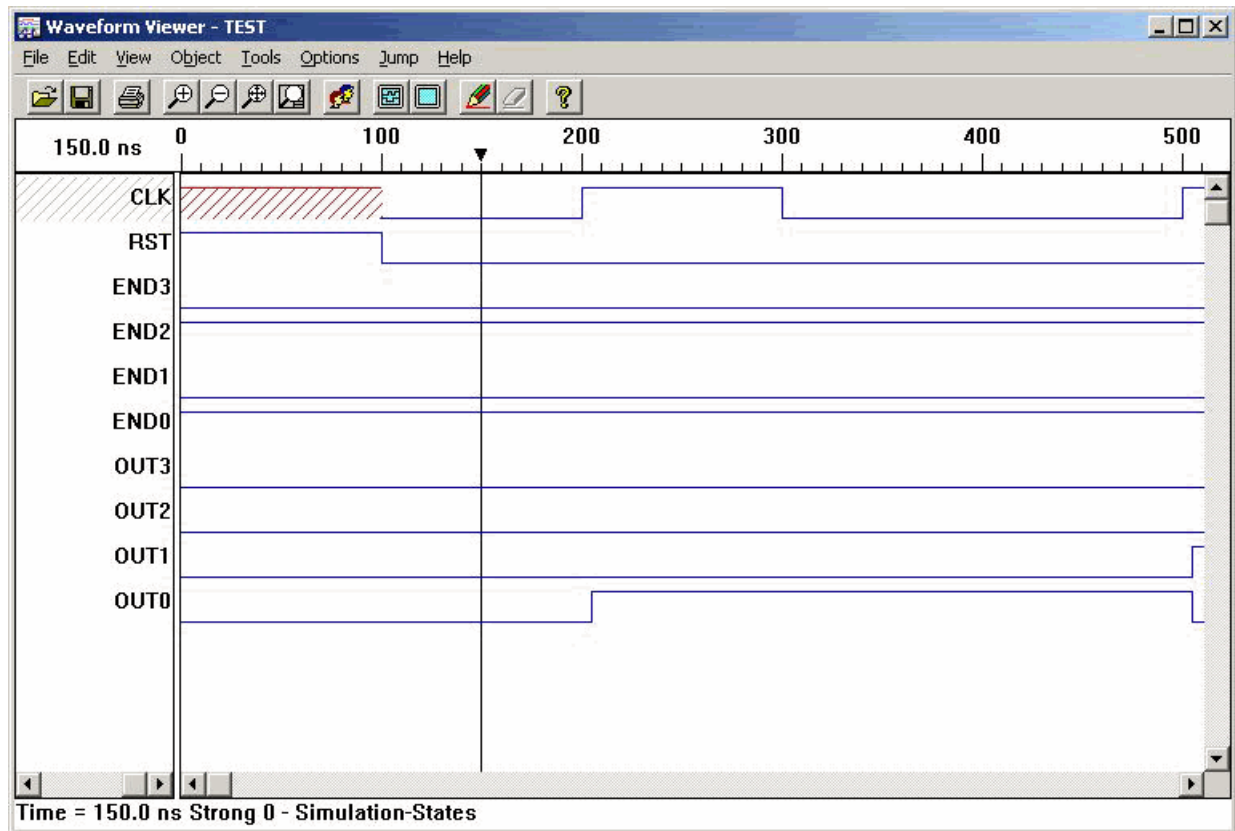
You can use the Waveform Viewer to measure the time difference between two events. In this task, you will measure the t_{CO} and see how it compares with the results of the Performance Analyst in a previous task.

To measure the difference between two events:

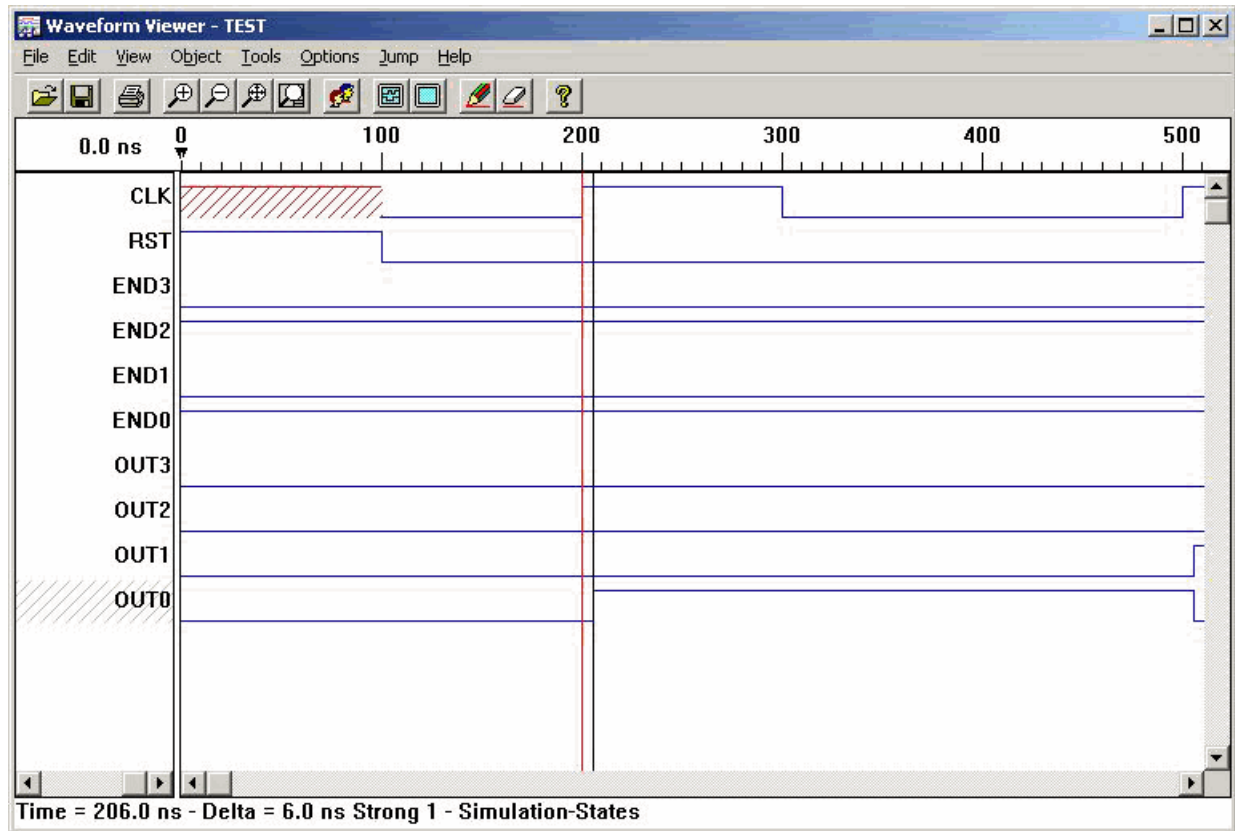
1. On the toolbar, click the **Zoom In** icon (). The cursor switches to a large "Z." Click in the waveform display area until the major scale is in 100-nanosecond increments, as shown in the horizontal timeline below the toolbar. Right-click to return to the regular cursor.
2. Choose **Jump > Time=0**.



3. Select the **CLK** signal by clicking the label in the waveform name area. Click the Query cursor at about time **150 ns**. This sets a vertical marker at that point.



4. Choose **Jump > Next Change**. The marker jumps to the time 200 ns.
5. Choose **Object > Place Marker** to set a "permanent" marker at this point.
6. Now select the **OUT0** signal.
7. Choose **Jump > Next Change**. The temporary marker jumps to time 204.4 ns. In the status bar, notice **Delta = 6.0 ns**. Remember that t_{CO} measured in the Performance Analyst was also 6.0 ns.

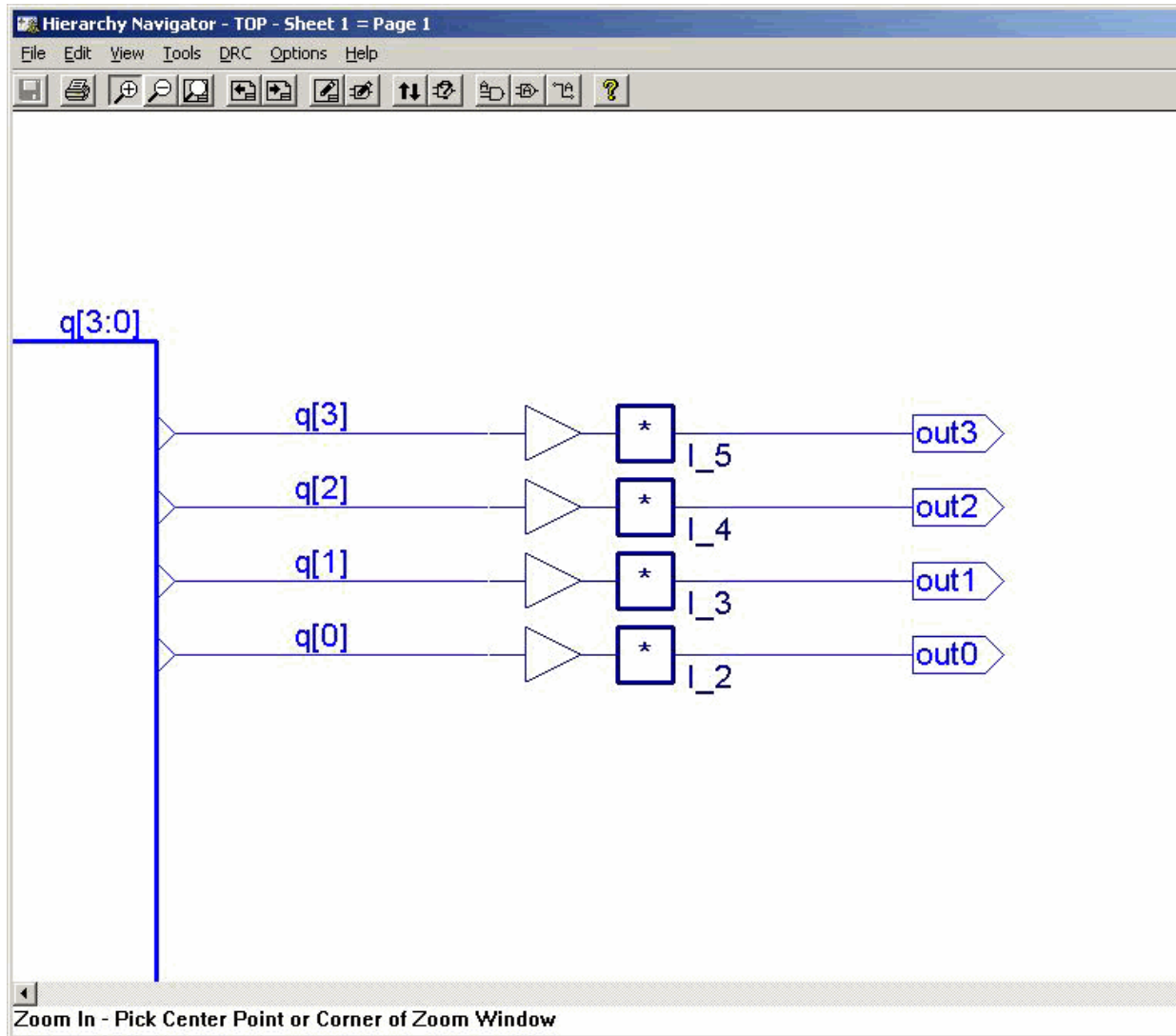


Task 4: Correlate Simulation Results by Cross-Probing

Cross-probing enables you to display simulation results on schematics and to add waveforms to the Waveform Viewer from a schematic. Crossing-probing also makes it easier to correlate simulation results with the sources in the design.

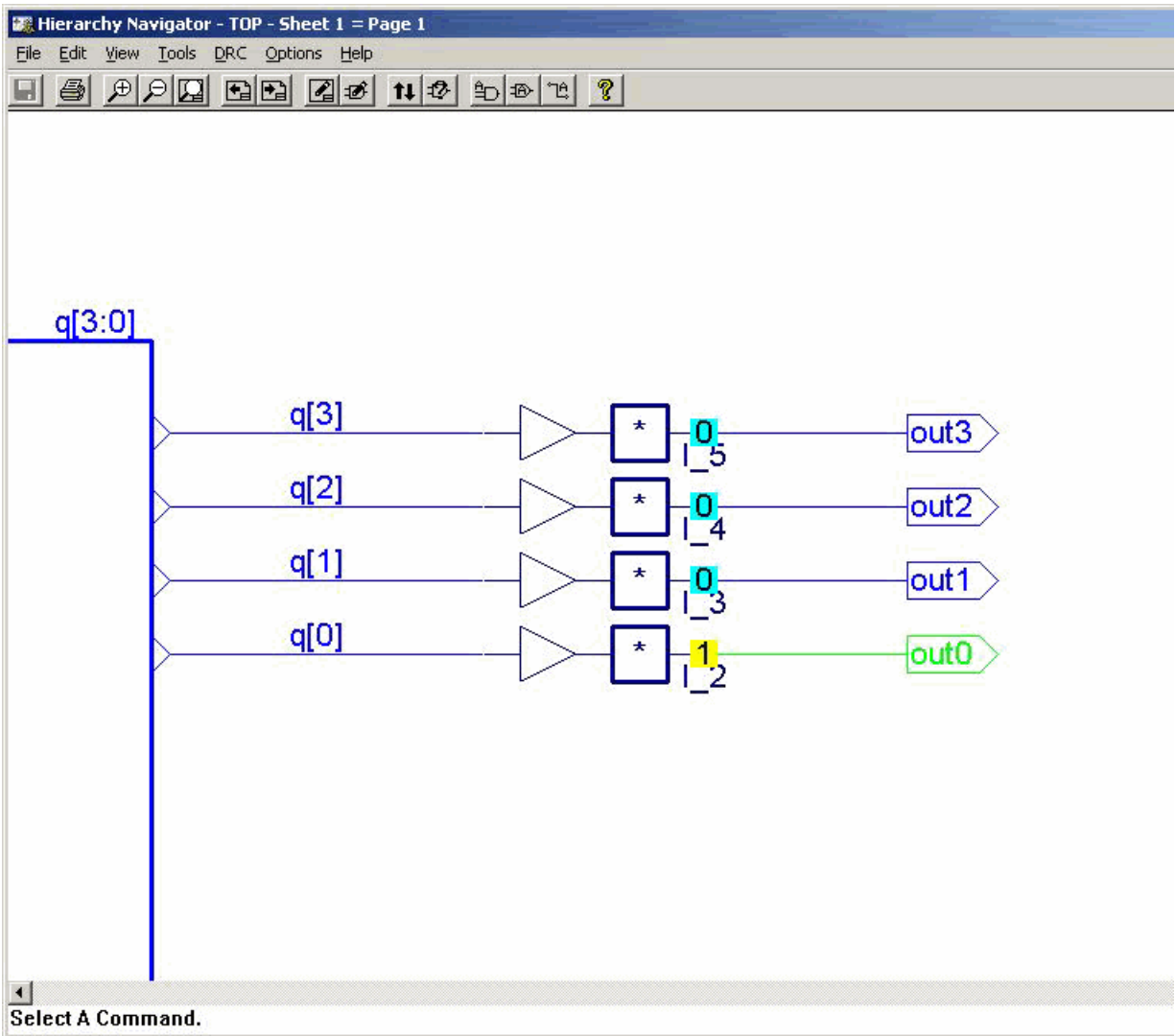
To cross-probe:

1. While the Waveform Viewer is running, select the **top** schematic source (`top.sch`) in the Project Navigator.
2. In the Processes for Current Source window, double-click **Navigate Hierarchy** to open the Hierarchy Navigator. Position it beside the Waveform Viewer.
3. In the Hierarchy Navigator, choose **View > Zoom In** and enlarge the view of the output pins on the far right of the schematic (`out0-3`).



4. In the Waveform Viewer, choose **Object > Hide Marker**.
5. Choose **Jump > Time=0**.
6. Select the output pins **out0-3** one at a time. As you do, notice that the corresponding pin in the Hierarchy Navigator changes color.
7. In the Waveform Viewer, select the **OUT0** signal name. Then choose **Jump > Next Change**. The time marker moves to Time=204.4 ns.

The logic values determined during simulation are displayed on the schematic loaded in the Hierarchy Navigator. Because you were at time = 0, all the outputs were low (0). However, after you jumped the time marker to the next transition, signal OUT0 changed to high (1).



8. Choose **Jump > Next Change** again to move the time marker to time 504.4.

Now you are on the transition where OUT0 goes back low and OUT1 goes high. As the cursor jumps to different points along the time line, the logic values on the schematic change to those for that simulation time.

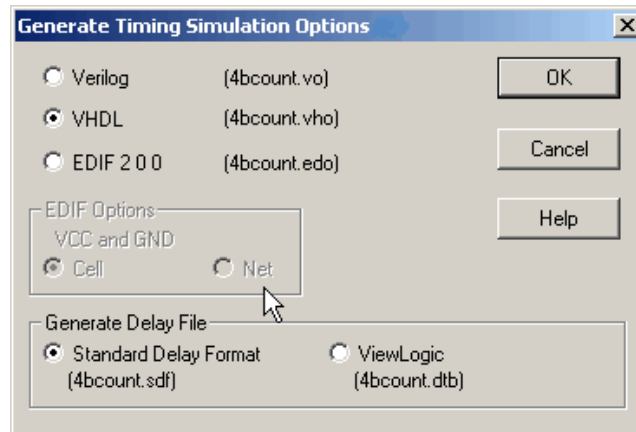
9. Close the Hierarchy Navigator, Waveform Viewer, and Lattice Logic Simulator.

Task 5: Timing Simulation for Third-Party Simulators

After you have fit the design, the ispLEVER software enables you export the netlist and delays for timing simulation. For netlist files, ispLEVER supports VHDL, EDIF, and Verilog formats. For timing delay files, ispLEVER supports the standard SDF and Viewlogic DTB timing formats.

To generate a third-party simulation netlist:

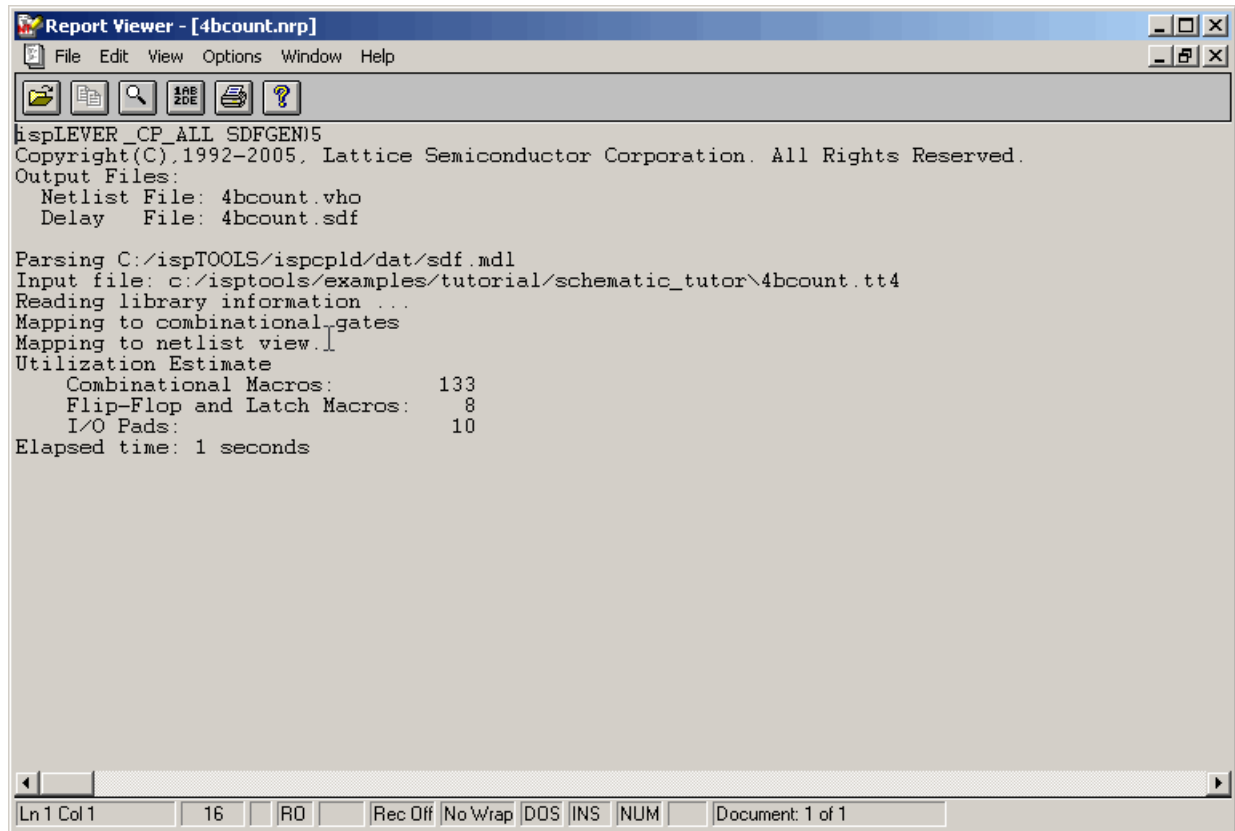
1. In the Project Navigator, choose **Tools > Generate Timing Simulation Options** to open the dialog box.



2. Depending on the timing simulator you are using, the ispLEVER software lets you choose a specific format for the output netlist and the timing delay file. Accept the format settings and click **OK** to close the dialog box.
3. In the Project Navigator Sources window, select the target device.
4. In the Project Navigator Processes window, double-click **Generate Timing Simulation Files**.

The ispLEVER software generates a report file, which contains a summary of the design as well as the name and type of netlist file and delay file. Your timing simulator can read these files.

5. Right-click on the **Report File** process and choose **View** to view the results generated from the Generate Timing Simulation Files process in the Report Viewer.



```
ispLEVER_CP_ALL SDFGEN)5
Copyright(C).1992-2005. Lattice Semiconductor Corporation. All Rights Reserved.
Output Files:
  Netlist File: 4bcount.vho
  Delay File: 4bcount.sdf

Parsing C:/ispTOOLS/ispcpd/dat/sdf.mdl
Input file: c:/ispTOOLS/examples/tutorial/schematic_tutor\4bcount.tt4
Reading library information ...
Mapping to combinational gates
Mapping to netlist view.
Utilization Estimate
  Combinational Macros:      133
  Flip-Flop and Latch Macros: 8
  I/O Pads:                  10
Elapsed time: 1 seconds
```

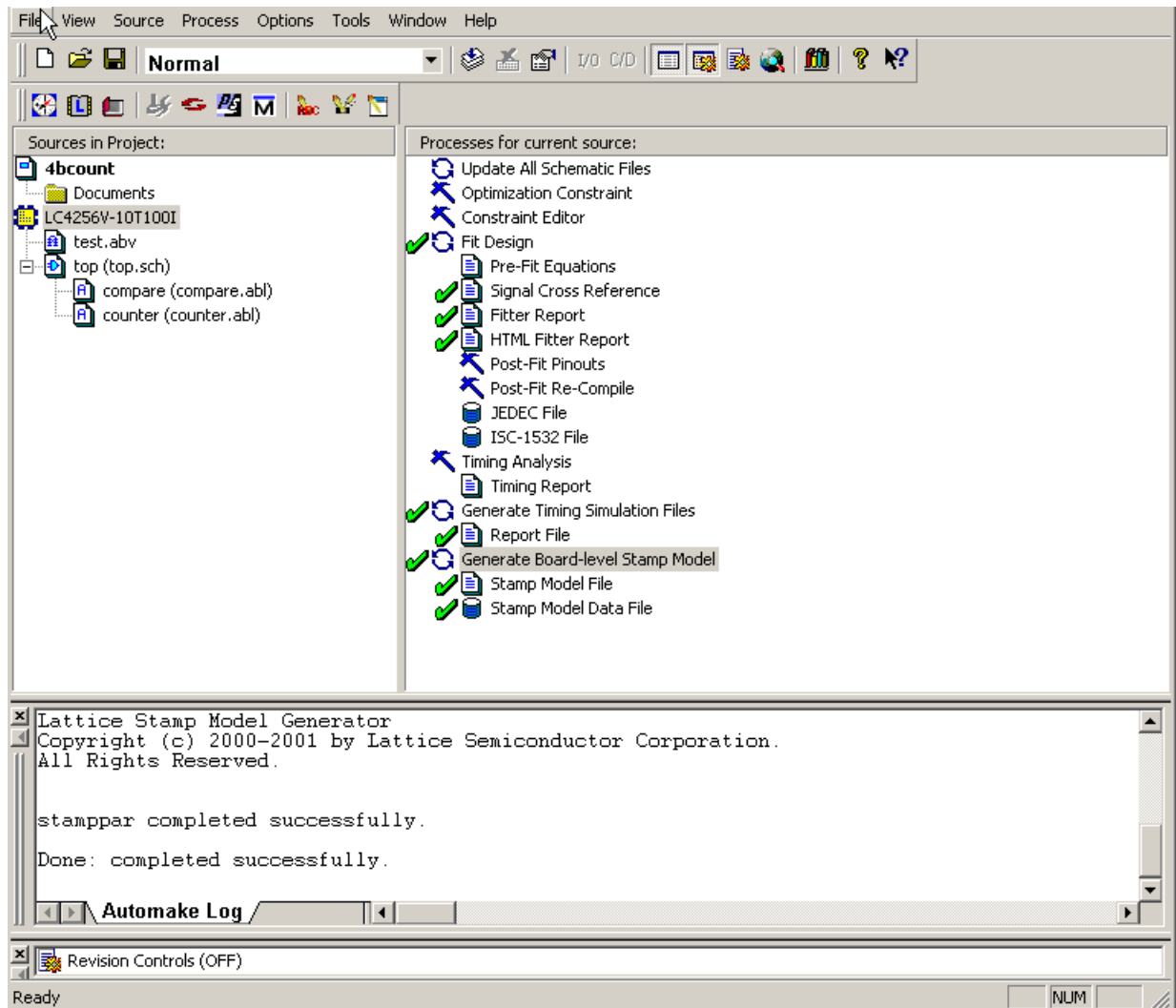
6. Close the Report Viewer when you are finished viewing the report.

Task 6: Board-Level Static Timing Analysis

You can use the Lattice Stamp Model Generator to build board-level stamp models of a design using any Lattice Semiconductor device for third-party board-level static timing analysis tools. A board-level static timing analysis tool enables you to manage large, high-performance designs while minimizing development time. With the stamp models, you accelerate board-level design.

To generate stamp model files:

1. In the Project Navigator, select the target device, and then double-click the **Generate Board-level Stamp Model** process. The software generates two files: the the Stamp Model file and the Stamp Model Data file.



2. Right-click on these file and choose **View** to open the files in the Report Viewer.
3. Close the Report Viewer when you are finished viewing the files.

Summary

You have finished the Schematic and ABEL-HDL Design tutorial. In this tutorial, you have learned how to do the following:

- ◆ Set up an ispLEVER schematic project.
- ◆ Create a top-level schematic source.
- ◆ Create and import ABEL-HDL sources into a project.
- ◆ Navigate a design.
- ◆ Import a test vector file and perform functional simulation.

- ◆ Set pin constraints, fit the design, and read the fitter report.
- ◆ Perform timing analysis and simulation and analyze the results.
- ◆ Correlate simulation results using cross-probing.
- ◆ Export the netlist and delays for timing simulation.
- ◆ Build board-level stamp models of a design.

Glossary

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

ABEL-HDL. ABEL-HDL is the ABEL Hardware Description Language, which expresses logic as equations, truth tables, and state machine descriptions.

bus. A bus is a named collection of two or more wires or other buses that carry closely associated signals in an electronic design. Attaching a bus tap to a single wire "promotes" the wire to a bus. In a schematic, a bus looks like a single wire but is displayed in a different color. (The color does not appear until you name the bus.) The individual signals within a bus can be tapped off at any point along the bus, as often as needed. (The Schematic Editor does not check to see if you are overloading the bus.)

bus tap. A bus tap is the point at which a signal enters or exits a bus.

cross-probing. Cross-probing is the process of displaying simulation results on schematics.

fitting. Fitting is the process of putting logic from a design into physical macrocell locations in the CPLD.

flip-flop. A flip-flop is a simple two-state logic buffer activated by a clock and fed by a single input working in combination with the clock. Its states are High and Low. When the clock goes High, the flip-flop works as a buffer as it outputs the value of the D input at the time the clock rises. The value is kept until the next clock cycle (rising clock edge). The output is not affected when the clock goes Low (falling clock edge).

functional simulation. Functional simulation is the process of identifying logic errors in a design before it is implemented in a Lattice device. Because timing information for the design is not available, the simulator tests the logic in the design using unit delays. Functional simulation is usually performed at the early stages of the design process. It gives information about the logic operation of the circuit. It does not provide any information about timing delays.

I/O marker. An I/O marker is an indicator that identifies a net name as a device input, output, or bidirectional signal. It establishes net polarity (direction of signal flow) and indicates that the net is externally accessible.

net. A net is (1) a logical connection between two or more symbol instance pins. After routing, the abstract concept of a net is transformed to a physical

connection called a wire. (2) An electrical connection between components or nets. It can also be a connection from a single component. It is the same as a wire or a signal.

schematic. A schematic is a drawing representing a design in terms of user and library components.

SDF. SDF is an industry-standard file format for specifying timing information. It is usually used for simulation.

Stamp. Stamp is a CPLD static timing analysis tool and modeling language that can be used as the board-level static timing analyzer. It enables you to obtain the results of static timing analysis and eliminate the timing errors in significantly less time for a board-level design. Its modes are mainly chip-level.

static timing analysis. Static timing analysis is the process of verifying circuit timing by totaling the propagation delays along paths between clocked or combinational elements in a circuit. The analysis can determine and report timing data such as the critical path, setup and hold time requirements, and the maximum frequency. The primary advantage of timing analysis is that it can be run at any time and requires no input test vectors, which can be very time-consuming and tedious to create. Although timing analysis does not give you a complete timing picture, it is an excellent way to quickly verify the speed of critical paths and identify performance bottlenecks.

test vectors. Test vectors are sets of input stimulus values and corresponding expected outputs that can be used with both functional and timing simulators.

timing simulation. Timing simulation is the process of obtaining detailed information about gate delays and worst-case circuit conditions after the design is implemented. Because the total delay of a complete circuit depends on the number of gates the signal sees and on the way the gates have been placed in the device, timing simulation can only be run after the design has been implemented.

Recommended Reference Materials

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

- ◆ ABEL-HDL Reference Manual
- ◆ ABEL Design Manual